The Transactor

The Tech/News Journal For Commodore Computers Vol. 4.

$2.50

ISSUE 02

Special:
Commodore 64
SID Sound Unleashed!

Butterfield Centerfold

commodore

Richvale Telecommunications

10610 BAYVIEW (Bayview Plaza)
RICHMOND HILL, ONTARIO, CANADA L4C 3N8
(416) 884-4165

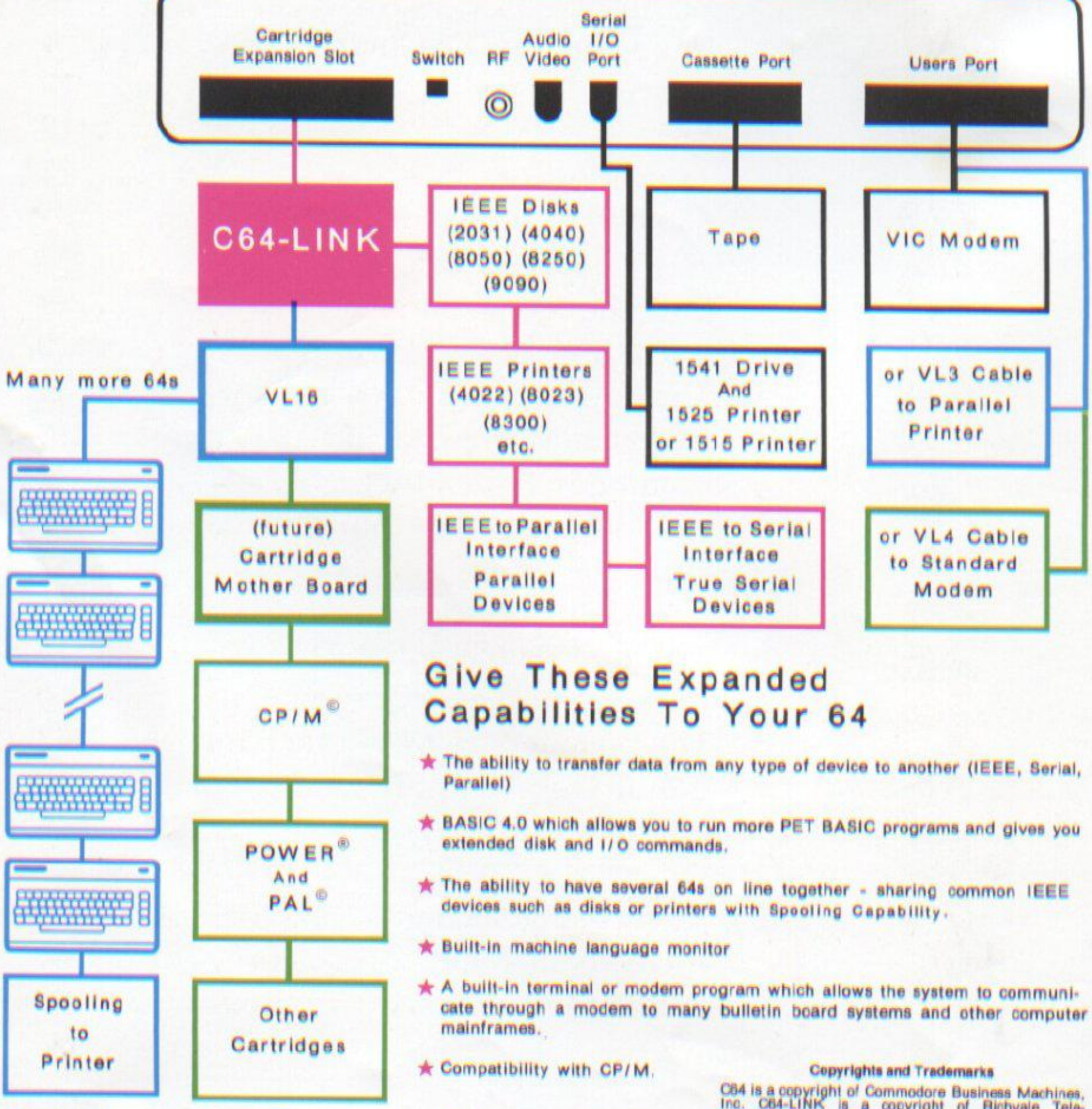
$185 Canadian

C64-LINK®

The Smart 64

Also available for VIC 20

RTC RTC



Give These Expanded Capabilities To Your 64

★ The ability to transfer data from any type of device to another (IEEE, Serial, Parallel)

★ BASIC 4.0 which allows you to run more PET BASIC programs and gives you extended disk and I/O commands.

★ The ability to have several 64s on line together - sharing common IEEE devices such as disks or printers with Spooling Capability.

★ Built-in machine language monitor

★ A built-in terminal or modem program which allows the system to communicate through a modem to many bulletin board systems and other computer mainframes.

★ Compatibility with CP/M.

Contact your local Commodore dealer or RTC.

Copyrights and Trademarks

C64 is a copyright of Commodore Business Machines, Inc. C64-LINK is a copyright of Richvale Telecommunications. CP/M is a registered trademark of Digital Research. POWER is a trademark of Professional Software. PAL is a copyright of Brad Templeton.

The Transactor

Volume 4 Issue 1

Editorial 3
News and New Products 4
Bits and Pieces 10
Bugz 18
Transbloopors 23
6551 ACIA Registers 24
The WordPro Book of Tricks 26
Bulletin Board Numbers 29
The MANAGER Column 30
The Four POKE Screen Marker 34
Determining Screen Size 36
Computer Technology 39
4040 Power/Error Indicator 40
More On Filing 42
Translator Utility 44
Making Friends With SID 46
C 64 Piano/Organ Program 50
The Commodore 64 Skiffle Band 53
Editor Link To BASIC–Aid 56
C 64 Joystick Ports 60

Advertising Section 61
Advertising Index 64
Next Issue 64

The Transactor
The Tech/News Journal For Commodore Computers

Editor
Karl J. H. Hildon

Editorial Assistant
Elizabeth Baillie

Contributing Writers
Greg Beaumont
Dave Berezowski
Jim Butterfield
F. Arthur Cochrane
Donna Green
Paul Higginbottom
Dave Hook
Eike Kaiser
Robert Lovelace
Bill MacLean
Chris Siebenmann
John Stoveken

Production
Attic Typesetting Ltd.

Printing
Squire & Carter Ltd.

Cover Art
Jim Jones, McLean Waite Advertising

Artwork
John Mostacci

Program Listings In The Transactor

All programs listed in The Transactor will appear as they would on your screen in Upper/Lower case mode. To clarify two potential character mix-ups, zeroes will appear as '0' and the letter "o" will of course be in lower case. Secondly, the lower case L ('l') has a flat top as opposed to the number 1 which has an angled top.

Many programs will contain reverse video characters that represent cursor movements, colours, or function keys. These will also be shown exactly as they would appear on your screen, but they're listed here for reference.

Occasionally programs will contain lines that show consecutive spaces. Often the number of spaces you insert will not be critical to correct operation of the program. When it is, the required number of spaces will be shown. For example:

print" flush right" – would be shown as – print" [space 10]flush right"

Cursor Characters For PET / CBM / VIC / 64

Down	– q	Insert	– T
Up	– Q	Delete	– t
Right	–]	Clear Scrn	– S
Left	– [Lft]	Home	– s
RVS	– r	STOP	– c
RVS Off	– R		

Colour Characters For VIC / 64

Black	– P	Orange	– A
White	– e	Brown	– U
Red	– Z	Lt. Red	– V
Cyan	– [Cyn]	Grey 1	– W
Purple	– [Pur]	Grey 2	– X
Green	– I	Lt. Green	– Y
Blue	– ←	Lt. Blue	– Z
Yellow	– [Yel]	Grey 3	– [Gr3]

Function Keys For VIC / 64

F1	– E	F5	– G
F2	– I	F6	– K
F3	– F	F7	– H
F4	– J	F8	– L

The Transactor is published bi-monthly, 6 times per year, by Canadian Micro Distributors, Limited. It is in no way connected with Commodore Business Machines Ltd. or Commodore Incorporated. Commodore and Commodore product names (PET, CBM, VIC, MAX, 64) are registered trademarks of Commodore Inc.

Volume 4 Subscriptions: Canada $15 Cdn
U.S.A. $15 US.
All other $18 US.

Second Class Mail
Permit Pending

Back issues are still available for Volumes 1, 2, & 3. Best of Volume 1: $10 Cdn., U.S.A. $10 US., all other $12 US. Best of Volume 2: $15 Cdn., U.S.A $17 US., all other $19 US. Volume 3: $15 Cdn., U.S.A $17 US., all other $19 US.

Volume 4 quantity orders: subtract 35% on orders of 10 or more.

Send all subscriptions to: The Transactor, Subscriptions Department, 500 Steeles Avenue, Milton, Ontario, Canada, L9T 3P7, 416 878 7277.

Want to advertise a product or service? Call or write for more information.

Editorial contributions are always welcome and will appear in the issue immediately following receipt. Preferred media is 2031, 4040, 8050, or 8250 diskettes with WordPro, WordCraft, Superscript, or SEQ text files. Program listings over 25 lines should be provided on disk or tape. Manuscripts should be typewritten, double spaced, with special characters or formats clearly marked. Photos of authors or equipment, and illustrations will be included with articles depending on quality. Diskettes, tapes and/or photos will be returned on request.

All material accepted becomes the property of The Transactor, until it is published. Once released, Authors may re-submit articles to other publications at their own discretion. Other magazines, of any nature, are invited to copy material from The Transactor, provided that credit is given to the Author (when applicable) AND The Transactor.

The opinions expressed in contributed articles are not necessarily those of The Transactor. Although accuracy is a major objective, The Transactor cannot assume liability for errors in articles or programs.

From The Editor's Desk

I'd like to take this opportunity to apologize for the delay in releasing our 1st and 2nd issues, but we now have some momentum built up and future releases should be fairly regular. However, I think you'll agree that anything worth doing right is worth a little extra effort.

Or a lot extra, as the case may be. Early in our planning stages we decided that program listings should also be typeset, the main reason for the delay. At the risk of bending your ear, I'd like to explain some of the technicalities of this very exact science.

First of all, I thought it might be quicker to find a typesetter that could accept data directly from the computer rather than re–keyboarding the articles. And I did. . . Attic Typesetting of Toronto (see the News section of Issue 01 for details). Getting the two machines to communicate was a snap, and in fact, the easiest task of all, as I was to find out later.

Once the text was in the typesetter editor, several other problems became apparent. The wordprocessor breaks lines of text at positions that are not necessarily at the same spots the typesetter would break at. Therefore, all carriage returns sent by my machine had to be replaced by spaces in the typesetter editor. Two carriage returns, however, indicate a paragraph break, and are replaced by a typesetter "End Paragraph" command. But what about two carriage returns that indicate extra spacing between blocks of text? This presented a new wrinkle that could only be solved one way. . . by learning the art of typesetting. Nate Redmon of Attic handed me 3, inch thick manuals and off I went.

After studying some 130 commands and variations, I came back and said, "Nate. . . I get the impression that several of these are not often used?". Nate's answer? A simple "No", followed by a delayed smile. "Uh huh, I see..", and away I went again.

Next came the special characters used in program listings. Symbols like <, >, +, =, ↑, and \ are not all available from certain type faces. This requires a command to change type faces, set the character, and return back to the previous typeface. Reverse field cursor graphics. . . well, explaining these would take about as long as the typesetter command string required to obtain them.

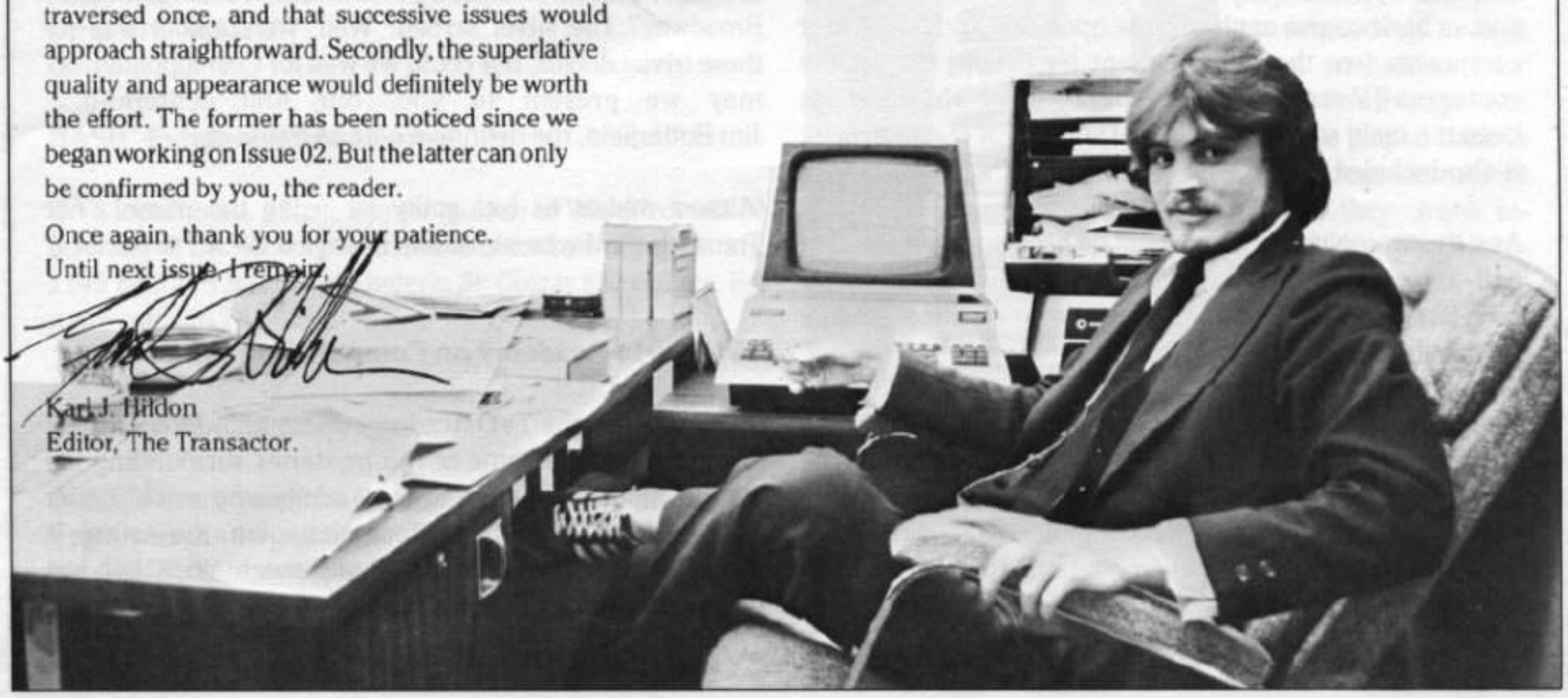
Fortunately most of this can be done automatically. The typesetter editor has a feature called "Translation Tables". Much like a giant search and replace, it allows a maximum of 300 different sequences of characters to be modified simultaneously. Building the translation table requires careful planning though, since character sequences that are subsets of other sequences must be placed lower in the list, else incorrect replacements occur. So far we've used up about half of the 300 available lines, and additions are frequent.

One other problem we managed to overcome was that of taking consecutive PRINT statements that contain graphics and "lining them up" to make them more legible for entering. An effective width table override was necessary, but since no such command exists, new width tables were compiled that give each character equal width, or "mono–width". The space character is not contained in the width table since it has a variable width. To defeat this, a special, rather cosmic sounding command is invoked called "Space Limit". This allows you to specify the minimum width used by a space and was set to the same value as our mono–width characters.

But all this was taking longer than I expected. The Transactor was getting later and later and I had to decide, "is it worth it. . . maybe I should just list everything on a regular printer using bracketed mnemonics to represent special characters, as in the past". After some discussion, we determined that the time spent here would only be traversed once, and that successive issues would approach straightforward. Secondly, the superlative quality and appearance would definitely be worth the effort. The former has been noticed since we began working on Issue 02. But the latter can only be confirmed by you, the reader.

Once again, thank you for your patience.
Until next issue, I remain,

Karl J. Hildon
Editor, The Transactor.



News and New Products

Jim Butterfield Centerfold

Special thanks to John Barr and Joan Gore of the Art Shoppe in Toronto for permission to use the premises and set-up assistance. "In all the world, there's only one Art Shoppe", and there's definitely only one Jim Butterfield. . . the two go together well, don't you think?

Jim has been working for the past several months as chief consultant for the *TVO Academy on Computers in Education*.. a basic course exploring the operation and function of computers. (see the next segment for details) Part of the course is a TV series called *Bits and Bytes* for which Jim has been the main source of technical content. A resource book is also included. . . Jim wrote most of that too.

As if this weren't enough, Jim will be host/guest expert on a half-hour follow-up show scheduled immediately after each Bits and Bytes program. At the time of writing, a name for the show had yet to be chosen.

Unlike Bits and Bytes, the second 1/2 hour will have an open format, off-the-cuff approach as opposed to a rigid script. In the first episode, the notorious Butterfield dismembers a perfectly innocent microcomputer, as studio participants witness this sinister perpetration. No word on the brand. . . a Radio Shack would be the most deserving. Nevertheless, it all sounds like great fun!

Coupled with the stacks of material Jim pumps out every month, his Machine Language course at George Brown College, TPUG and other club meetings, presentations at conferences world wide, consulting for CBM, and the hundreds of phone calls he gets every week, his new TV role should make Jim one busy guy!

Is this the birth of a star? Where will it lead? Stratford? Broadway? The silver screen? Well, we couldn't wait for these trivial details, nor could we wait for Cosmopolitan. So, may we present to you, our first centerfold. . . Jim Butterfield, the definitive *human* transactor!

(Almost makes us feel guilty for using the name, "*The* Transactor". Maybe we should change it to "*A* Transactor")

TVOntario Academy on Computers in Education

As mentioned, the TVO Academy on Computers is a course designed to solve some of the mysteries surrounding the microcomputer. Catering to busy adults who would rather learn at home than travel to a classroom, the course is actually a package deal that includes much more than just the TV show. For $59 you get a pile of stuff!

- A 64 page Participant's Guide
- A 64 page Resource Book (by J.B.)
- The Academy Newsletter, to be published 3 times during the series
- A 40 page Educational Applications Handbook
- A Hands–On Beginner's Manual
- Related Sample Software (cassette tape)

For $53 dollars you get everything but the last 2 items. Either way, it's a bargain if this is what you've been waiting for. Also consider that community college night courses start around $55 with books extra.

The course begins with the first of 12 shows airing Wednesday February 16th at 9:00 pm. on your local TVOntario channel (19 in T.O.). Each show is repeated the following Saturday at 12:30 pm. Some topics will include languages like BASIC, LOGO and PILOT, filing techniques, modems and network communications, graphics and music, wordprocessing, spreadsheet programs, and the future of Computer Assisted Instruction (CAI).

Bits and Bytes is hosted by Billy Van and Luba Goy. Van plays the part of the anxious–to–learn, inquisitive "average" viewer. Luba Goy is the course instructress. The pair (both from comedy backgrounds) should have no trouble maintaining a light, friendly, and entertaining atmosphere.

Once again, Jim Butterfield follows each Bits and Bytes. No mention of this show is made in the brochure, but it will surely be considered an added bonus. TVO is also working on a "buddy system", where registrants will be informed of others in their immediate area.

For more information on enrollment, call Ariel Lang Loughridge at TVO, 2180 Yonge Street, Toronto. The number is 416–484–2607.

BASIC Fundamentals Course

On a local note. . . Steve Punter (author of WordPro) will be teaching a BASIC course at the Etobicoke Public Library, 1745 Eglinton Ave. W. at Dufferin St. Cost is $30 dollars. For more call 416 781 5208.

January 83 CES Las Vegas

Just like every year, the Consumer Electronic Show in Las Vegas was the largest so far. Somewhere in the neighborhood of 70,000 came to satisfy their curiosity. . . an impressive follow–up to the largest ever Comdex (Computer Dealers Exhibition) last November. Around 50,000 showed up for this one which is supposedly off limits to the public.

Several new products made their debut at Comdex, several more at CES. Commodore tends to save their new stuff until CES. This year there were 5 items premiered (or should I say 4 and a half). Let's get to these first.

The SX–100

Basically, a Commodore 64 in a different casing. The unit, designed to be portable, is Commodore's answer to the Osbourne. It's smaller than the Osbourne (weighing in at 18 lbs.) and comes with 1 or 2 disk drives and a 5 1/4 inch screen. No details yet on availability or price, probably around the $1500 US. mark.

Commodore 64 Add–On Keyboard

Only the prototype (below) of this product was shown. It plugs into the cartridge slot of the 64 and should also work with the MAX when it comes. The connector is also a chassis for three extra SID chips and some Parallel Interface Adapters (PIA). Coupled with the on–board SID of the 64, the combo will allow 12 voices to be played simultaneously.

Paul Higginbottom of Commodore Canada is writing software for the keyboard that will allow you to assign voices to pre–defined string, wind instruments and/or percussion accompaniment. Pieces can be recorded into memory and you can play along with your own recordings or save them on disk for later use. Recordings can also be played back in a different instrument than they were recorded in. Ten instrument presets are built into the program that can be altered to suit. The keys are not velocity sensitive, but at $300 US. (unconfirmed), this product is still a excellent deal.

Availability has been announced for May 83 with a possible VIC-20 equivalent to follow.

CBM Speech Synthesizer

This one also plugs into the C64 cartridge slot. Capabilities include built-in software for direct speech output and phrase programming, recording, and playback. No availability or price as yet.

1520 Printer Plotter

This was first shown at Comdex in '82 but displayed with more emphasis at CES. It has four colours (red, green, blue and black) and uses regular paper up to about 6 inches wide. Print speed is 14 cps with a 0.2 mm. accuracy. Alpha-numeric output can be 20, 40, or 80 columns, and for 200 bucks US., it should do ok. Available now but expect back orders.

The HHC-4

This is the "half product". "HHC" stands for Hand Held Computer. It has a full QWERTY keyboard (with keyword "burst" keys), a numeric pad, comes with 4K of RAM, expandable to 12K. The machine is actually a Toshiba product with a Commodore label and looks a lot like the one Radio Shack sells. Plugged into an interface adapter, the HHC-4 can be connected to a cassette, dot matric printer, TV or monitor, and can even talk with the VIC-20 or C64. But the processor is slower than snail doo. Screen scrolling is abismal. Rumour is that Commodore will either replace the guts of the machine with their own design or make a whole new one from scratch, but don't expect to see this one until it's improved. Once done, it would give the R.S. equivalent a lot of trouble, especially since CBM will undoubtedly beat their price.

Commodore Colour Monitor

This has been around a while now. The $300 US. price tag makes it pretty attractive, especially with the special "luminance" input for use with the 64. Call your dealer for availability.

VICModems Come To Canada!

Commodore has finally started producing the necessary ingredient to make VICModems work in the "great white north". Unlike Canada, U.S. telephone handsets are connected to the dial housing via a cord that can be unplugged from both pieces. This makes set-up simple. The handset is merely detached and the cord is re-routed to the VICModem.

The extra component for Canadian use is like a "Y" connector made from Ma Bell style jacks. This plugs into the VICModem. From there, one side goes to the telephone, the other to the wall outlet.

Expect to see VICModems at your dealer sometime in the spring or summer. Price in Canada will be under $200.

Dual Drives Take New Shape

It may not be long before CBM dual drives start appearing in a new housing. Quite simply, it looks like a VIC single drive housing, but double the width. This is pure speculation on my part – they could be limited to European distribution. Is this the reason for the 4040 shortage?

The Commodore Space Shuttle?

Not quite. . . maybe next year. But with over 60 complete Commodore systems at JFK Space Center, they certainly deserve mention. NASA officials estimate that the systems in use have improved the overall efficiency of their documentation department by 5 fold. Other applications include equipment and employee tracking, job costing, status reports, and tracking hazardous waste matter to ensure safety. Although not as dramatic as the Anik-C satellite and Spar's Canadarm, we thought you might like to know.

Turbo 1541 From BMB CompuScience

Turbo 1541 is a hardware interface for use with the VIC 1541 Single Disk Drive. Designed to overcome the speed limitations of the serial bus, the Turbo 1541 is a parallel interface that connects to the User Port. The other end is a PC board that plugs into the microprocessor socket of the 1541 drive, thus bypassing the serial bus altogether.

Versions will be available for both VIC-20 and the Commo-

dore 64. The device is "software transparent" which means it will work with any software that performs filing operations or dynamic loads from the 1541 drive. Speed improvements approach that of the 4040 dual drive. Disconnect it and your 1541 is back to normal (ie. no permanent hardware modifications required).

Turbo 1541 should be available this summer at a price around $100.

Light Pen From Madison Computer

Madison Computer's newest product is The Light Pen, designed for use with VIC-20 and Commodore 64 computers. Unlike other light pens, this one allows full use of your computer with the pen in operation. The Light Pen always knows where it is on the screen - individual dots can be selected quickly without the need for blinking points.

The pen connects via the joystick port and is housed in an attractive oak stand that also holds the intensity control dial. Between the pen and the stand is a lightweight four foot cord to permit freedom of movement. The pen itself contains a photo diode which detects the scanning signal on a TV set or monitor.

Also includes is a complete manual and sample programs. For information, call or write:

Madison Computer
1825 Monroe
Madison, WI, 53711
608 255 5552

Another recent Madison product is The DeskTopper; an attractive oak stand for your VIC-20 or C64. It has room for one or two single drives, diskettes, manuals, and neatly hides that ugly mess of connector cables. Price: $49.95 US. plus shipping.

PromQueen From Gloucester Computer Bus

The PromQueen is a low-cost programmers that lets the user write machine code programs and burn them into EPROMs quickly and easily. The VIC-20 version is available now (suggested retail $199 US.) with the C64 version well on its way. 4K of RAM is included that allows for address dependent testing prior to burn-in.

The package comes with Hexkit 1.0; a software editor and debugging system. Machine code is usually regarded as very difficult to learn, requiring a long, arduous, and time-consuming study process. Hexkit 1.0 helps relieve some of the pain and frustration often experienced when perfected machine language programs.

Other features include a top quality zero insertion force (ZIF) socket that will accept 2716, 2732, and 2732A EPROMs. PromQueen is double buffered to offer full isolation from the host computer at the flick of a switch. The ZIF socket is also buffered, allowing gang programming of up to 10 EPROMs simultaneously. It plugs right into a stock VIC-20; no additional support devices or power supply is required.

For more information, call:

IBC Distribution
4047 Cambie St.
Vancouver, BC, V5Z 2X9
604 879 7812

Arbutus Total Soft
4202 Meridian, Suite 214
Bellingham, WA, 98226
206 733 0404

Or: Wally Ralston
Gloucester Computer Bus
6 Brooks Rd.
Gloucester, MA, 01930
617 283 7719

Microbuffer From Practical Peripherals

Microbuffer is an intelligent interface card that fits between your computer and printer. It takes printer-intended data and stores it in its own 32K of memory, which is expandable to 256K. Microbuffer will accept this data as fast as your machine can send it. While the buffer is filling up, it simultaneously empties the data to your printer at its slower speed. Once all data has been sent, your computer is now available for further use. . . ie. no waiting for long program listings to finish printing before continuing with your next task.

Microbuffer will speed up any program that requires printing. Once output is sent into its memory, the data will trickle out to the printer while you set up for your next report. In most cases memory will not fill entirely. If it should, upon receiving the trailing 32K (max 256K) of your report, control is released to the keyboard. Depending on printer speed, you could have up to an hour for working on non-printer operations.

The control panel includes a "pause" button, a "clear" button, and a "copy" button for duplicating memory contents from the last output.

Two versions are available; one for RS–232 serial, another for Centronics parallel, making it compatible with Epson, Centronics, NEC, C.Itoh, Qume, Diablo (CBM 8300P), Anadex, and IDS Paper Tigers. Prices start at $299 US. with 64K expansion modules at $179. Want more? Call:

Practical Peripherals, Inc.
31245 La Baya Drive
Westlake Village, CA, 91362
213 991 8200

Formaster Diskette Duplicators

Are you a software distributor using stacks of dual floppies to make your disk duplicates? If you're delivering 8050 or 8250 format diskettes, you know how long it takes to make a single copy.

Formaster now offers an alternative. Their dual drive disk duplicators will copy up to 326 disks per hour with the automatic loading option. On type 4040 disks, that's about 11 seconds per copy! (20 s. for 8050, 33 s. for 8250, a ratio of about 22:1) Features include ease of operation, error checking, and piracy protection. Call or write:

Formaster
2102 Ringwood Avenue
San Jose, CA, 95131
408 942 1771

Vectrix Hi–Res Colour Graphics Machine

The Vectrix is an RS 232 device that fits between your computer and colour monitor (composite or RGB). It comes in two models; the VX128 and VX384. Both machines offer a resolution of 672 × 480 individually addressable pixels. The internal processor is an Intel 8088 running at 5 MHz with a pixel drawing time of 1600 nanoseconds. Software features include point, line, arc, solid and pattern filled polygons in 2D or 3D. Commands in 3D allow rotation, scaling, translation, perspective, clipping, and viewport

The VX128 has 128K of RAM, offering 3 bit planes for a total of 8 possible colours per pixel with 8 colours to choose from. Price: $1995 US.

The VX384 has (get this) 384K of RAM for a total of 9 bit planes. With 9 bit planes you have a possible 16.8 million (yes, million!) colours to choose from. At any one time, one of 512 of these colours can be directed to any pixel on the screen for truly superlative CRT graphics capabilities. This model has some extra commands: slant, magnification (zoom), and programmable character sets and spacing parameters. The price is $3995.00 US. so they're not for the average user. For more on both products:

Vectrix Corporation
700 Battleground Avenue
Greensboro, NC, 27401
800 334 8181
919 272 3478

Micrograph From International Micro Video

Micrograph is a full featured intelligent ink plotter for presenting breakdowns of statistical data in graphic form. You simply enter the data, select the type of graph best suited, and choose the colours you want. Micrograph does the necessary calculations, sets the scales, and will even draw on film or acetate for overhead projector presentations.

Graph types include Pie charts, Bar and Gatt graphs, Line graphs, and Rectangle charts in up to 8 different colours. Controllable plotting capabilities are included for other types of output. Alpha–numerics are can be positioned anywhere on the drawing surface. For information:

International Micro Video
1606 8th Street
Niagara Falls, NY, 14305
716 285 4855

Pacific Coast Software C64 Programs

Pacsoft, a California custom programming firm for CBM equipment since 1980, is now retailing software for the C64. Among them is a wordprocessor for $74.95, a programmers toolkit package for $69.95, an assembler for $59.95, and a disk filing base system for $39.95. They also have several new games for the 64.

Pacific Coast Software
3220 S. Brea Canyon Road
Diamond Bar, CA, 91765
714 594 8210

PetSpeed 64 Now Available

PetSpeed, the compiler program for Commodore computers, is now available in a version for the C64. See your PetSpeed dealer for details.

Looking For Software?

The Whole PET Catalog has been available for several months now. In it you'll find over 500 honest and independent reviews of software and hardware products for all Commodore computers. If you're looking to acquire more software for your machines, check out this book. Although a particular program may not be included in the reviews, other similar packages might be there for comparison. Contact your dealer or call:

Midnite Software Gazette
635 Maple
Mt. Zion, IL, 62549
217 864 5320

The Torpet
381 Lawrence Ave. W.
Toronto, Ont., M5M 1B9
416 782 9252

Access Telidon From Commodore Computers

Batteries Included, a Toronto CBM dealer, is now carrying NORPAK videotex products that make Commodore computers compatible with the Telidon network

The package actually enable any micro to work with a decoder as a videotex graphics display generator, thus offering an affordable alternative to the many expensive hardware systems required to connect with Telidon and other videotex networks.

Batteries Included
71 McCaul Street
Toronto, Ont., M5T 2X7
416 596 1405

Independent Commodore Dealers' Association

The ICDA was formed in October 1982 to assist Commodore dealers in two main areas with others to follow in coming months:

Purchasing

By utilizing central buying, the dealers will be able to purchase both hardware and software at considerable savings. Negotiations are now underway with monitor and card reader companies with expected savings of at least 50%. In the software area, an agreement has been signed with a data base company which will provide dealers with discounts approaching 70%.

Advertising

Centralizing advertising and utilizing co-op funds will provide 30% to 40% discounts on radio, TV, etc. ICDA will help you combat the department store campaigns.

In the future, we will be offering many other new services: an 800 hot line for technical assistance, low cost insurance program, etc.

We believe the Association can be self-sustaining with low yearly dues and an override on product sales.

Charter memberships are available for $25 now and guaranteed annual dues of $75/year starting in March '83. All new members signing up after January '83 will pay $100/year.

Memberships are restricted to Commodore dealers only and are not open to the mass merchandising retailers. For information:

ICDA
1774 N. Glassell Street
Orange, CA, 92665
714 921 0230

New Newspaper From ETI

ETI Magazine, a well known world-wide electronics mag, is now publishing a newspaper called Teaching Electronics & Computing. As the name suggests, the tabloid is designed specifically with educational institutes in mind. The paper prints all the latest news and events, experiences of other schools and teachers, plus lots of technical material that often has no copyright restrictions (good for classroom distribution).

Published 9 times per year for $12.00 Cdn. ($15 in the US.)

If you're a teacher, it's FREE! (same for school boards).

Teaching Electronics & Computing
25 Overlea Blvd., Unit 6
Toronto, Ont., M4H 1B1
416 423 3262

Bits and Pieces

Got some interesting bit of info you'd like to share?. . . a POKE, a screen dazzler, a neat trick, or some other anomaly? Send it in! We'll be glad to print it! Ed.

The Transactor?

Since the release of our latest "new format" Transactor, we've had a lot of new interest from a lot of different people,. . . and for different reasons. But it seems that one of the questions most often asked is, "Why is it called The Transactor?".

Back in the olden days, Commodore's very first micro to hit the market, as most of you will remember, was called the **PET**; an acronym for **P**ersonal **E**lectronic **T**ransactor. Also, the word "transactor" was defined in a dictionary somewhere (that escapes my recollection) as "a vehicle or device for transferring information from one place to another". This was also the basis of our new masthead, designed to represent outward motion from a centre, but retain the familiar border that's the same shape as the stickers put on early machines.

So, influenced by these two facts, the name "The Transactor" was created. But enough nostalgic reminiscing and sentimental memorabilia, let's have some fun.

FTOUTSM

The name, although rather hard to classify as an acronym, means "For Those Of Us That Smoke M_____". We'll let you fill in the blank. This short one–liner, by Benny Pruden of Wayne, Pennsylvania, should get an award for variety of display.

```
1 c=32 : for n=1 to 41 : c=192-c : for a=0 to n :
  for b=32768+a to 34768 step n : poke b, c : next b, a, n
```

After entering RUN, the program will appear to do nothing. . . but don't hit your STOP key. . . it just needs to warm up. Once it gets going I think you'll agree. . . "not bad for a one–liner, eh?"

The program could be sped up slightly by substituting the literal numerics for variables which would be initialized on line 0. In fact, any BASIC program on any Commodore machine will run faster by using floating point variables to represent your constants (integer variables are not as fast*). When BASIC runs into a literal numeric such as "32768", it must first interpret each character and then convert it to floating point in the "Floating Point Accumulator" (see any memory map). During this, time is also spent on error checking. (* Integer variables aren't as fast because they also need to be converted to floating point) But BASIC just loves floating point variables. The value is looked up in the simple variables table, transferred into the F.P. Accumulator, and execution proceeds. Not only that, but if the same constant

is used in several places, you'll save on memory too. For a good example, see Paul Higginbottom's Un-Assembler program in Volume 4, Issue #01. Due to the nature of his program, Paul defines nearly every variable at the start which improves the run-time considerably.

Getting back to our one-liner. . . it's written for the 8032 so 4032, VIC-20, and 64 users will need to make changes.

4032 : change 34768 to 33768.
VIC-20 : change 32768 to 4096 or 7680* and
change 33768 to 4602 or 8186*
* use second value with 8K (or more) memory expanders.
C64 : change 32768 to 1024 and
change 34768 to 2024

Mind Twister / Brain Bender

This one will dazzle you! It's the handywork of John Stoveken in Milton Ontario. In fact, don't look at it too long or your grey matter will turn three shades of purple-ish orange and ooz out your nose onto your keyboard, and make a real mess.

The program simply fills the screen with characters and then starts playing with the Upper/Lower case register at decimal 59468. By tapping any key, Upper/Lower case mode is toggled at different rates. . . 256 in all! Hit a "1" to end the routine.

The program, like some others we've presented here in Bits & Pieces, is so fast that the video beam can't keep up with the changing display, thus producing the truly weird effects.

It's designed to work on 8032's; BASIC 2.0 and Fat Forty users will have no trouble making it work, but the results will differ. VIC-20 and C64 users will, once again, need changes; two to the screen start and end addresses and another for the Upper/Lower case control register (59468 on PET/CBMs). The code is relocatable so it will work wherever you have memory.

The BASIC loader that follows is all you need to get it going (or rather all it needs to get you going). RUN it and enter:

SYS 20480

The number "**1**" shown in **bold** on line 30 is the base character used to fill the screen. For different effects, try changing this to 193, 223, 255, or your choice. You "need-to-know-what-makes-it-tick" machine code addicts will find the source code following the loader.

```
10 for j=20480 to 20539 : read x : poke j, x : next : end
20 data 169, 128, 133,   1, 169,   0, 133,   0
30 data 168, 169,   1, 145,   0, 200, 192,   0
40 data 208, 247, 230,   1, 165,   1, 201, 136
50 data 208, 239, 169,  12, 141,  76, 232, 173
60 data  76, 232,  73,   2, 141,  76, 232, 165
70 data 151, 201, 255, 240,   7, 201,  49, 208
80 data   1,  96, 198,   2, 166,   2, 202, 208
90 data 253,  76,  31,  80
```

```
        *=   $5000
        lda  #$80     ;screen start
        sta  $01
        lda  #$00
        sta  $00
        tay
loop    lda  #$01     ;base char.
sta     ($00),y
        iny
        cpy  #$00
        bne  loop
        inc  $01
        lda  $01
        cmp  #$88     ;screen full?
        bne  loop
        lda  #$0c
        sta  $e84c    ;set graphic
loop2   lda  $e84c
        eor  #$02     ;toggle to
        sta  $e84c    ;text mode
        lda  $97      ;key pressed?
        cmp  #$ff
        beq  lop      ;no, do delay
        cmp  #$31     ;yes, is it a 1?
        bne  next     ;no, change delay
        rts           ;yes, end
next    dec  $02      ;alter delay
lop     ldx  $02      ;and do one
lop1    dex
        bne  lop1
        jmp  loop2
        .end
```

Loading C64 Programs On PET/CBMs

BASIC programs in PET/CBMs commonly start at $0401. But C64 BASIC programs start at memory location $0801 (remember, there is always a "zero" in the byte preceding BASIC text space). So Commodore, in their infinite wisdom, decided that the 64 would have the capability to re-locate PET/CBM programs. They LOAD right where the 64 wants them with all chain pointers adjusted perfectly.

Not so in the reverse situation! PET/CBMs didn't know there was going to be a C64. . . so programs written on the 64 will LOAD into the PET at address $0801. YUK!

There are actually two solutions to this one. First, we could move the Start–of–BASIC pointer in the PET "up" to $0801. Then a zero must be placed at $0800 to keep the operating system happy. A LIST would display your program ok, but SAVE or DSAVE would reep havoc! On PET/CBMs, SAVE always starts at $0401. This would store 1K of "fore–junk" before reaching the start of the actual program. Take this file over to the 64 and it will try and load this fore–junk, leaving memory full of organized hodgepodge. The only way around it is to use the M.L.Monitor to store the program from the PET, which is also painful because we need to find the end–of–program address for the .S command.

So forget all this. . . here's a much easier way (thanks to James Whitewood, Milton Ontario):

1. Type: NEW <RETURN>
2. LOAD the C64 program into your PET/CBM.
   A "LIST" at this point should result in "READY."
3. Add the following line of BASIC:

   0 rem <RETURN>

4. Now enter:

   poke 1026, 8 <RETURN>

5. Delete line 0 by typing a 0 and <RETURN>
6. Type "LIST" and your program will be there!

A brief explanation. In Step 1, the PET thinks no program exists because the first thing in memory is an end–of–BASIC *marker* (2 consecutive zeroes).

Step 2 loads the 64 program into the PET and sets the End–Of–BASIC *pointer*. PET now thinks a program exists in memory, but the end marker is still down at $0401, so LIST will never get past this point.

In Step 3, all memory between $0401 and the End–Of–BASIC pointer (which is pointing at the end of the 64 program) is moved up to make room for "0 rem".

The forward chain pointer for Line 0 is now pointing at $0407, the end marker. The first byte of the 64 program now lies at $0807. By altering *only* the high order byte of the chain pointer (Step 4), Line 0 is "linked" to the program residing at $0807. The low order byte need not be adjusted since it will be the same.

Step 5 deletes Line 0. The PET treats this like any other delete. The space occupied by Line 0 is reclaimed by moving the Lines above down. Line 0 is effectively "squeezed out". But since Line 0 was the first Line, the new first Line will be that of the 64 program. Presto! The 64 program is right where the PET/CBM is most comfortable with it.

Wouldn't it be nice if everything were so simple?

Cheating A Syntax Error

If you're extremely lazy like me, you probably take any chance to make programming more effortless. For example. The RUN command is three entire key presses followed by a Return. And you don't always hit them right. How many of you have entered "RUIN", "RUB" or "RUM". Argh! Right?

Well fellow short–cutters, Wayne Garvin of Toronto has this one for us. The keys Shift and RUN/STOP do the auto LOAD and RUN sequence. Some machines want to load from disk, others from tape, but every machine, back to the original PET, has this feature.

But try this! Type a letter (eg. "k") and then hit Shift–RUN/STOP. A ?Syntax Error will result and the LOAD is ignored. However, the characters R–U–N and Return are still in the keyboard buffer. These will be honoured as if YOU entered them, and your programs begins!

Just don't hit a number key first or the LOAD command will be entered on a program line which might mutilate existing code.

This little trick is convenient.. it means I'll see my program errors that much faster.

More Key Combos

If you have an 8032, 8096 or a SuperPET, you've probably used the ":" key to pause scrolling when LISTing a program or a Catalog. Try this. LIST a program and hit the ":" key; the listing stops at the bottom of the screen. Now, with your forefinger on the ":", use your pinky to press the RUN/STOP key; scrolling resumes. To pause, release RUN/STOP; to abort, release the ":".

Another I use often is "Shift, RVS, A & L". Hit Shift first, then RVS, and press the A and L keys together. This combination does a line insert. . . handy for inserting lines into programs visually. The screen will scroll down from the cursor line and you fill in the gap.

Out Of Memory Error?

```
10 gosub 20
20 goto 10
```

The problem is obvious. As line 10 calls line 20, line 20 routes to line 10, and around we go again. The "return" information placed on the stack by Gosub never gets removed. As more and more piles up, eventually the stack overflows and the ?Out Of Memory Error is displayed.

FOR-NEXT loops will also do it to you. If you start a FOR-NEXT loop and jump out of the loop with GOTO, information is left on the stack waiting for the NEXT statement to come along and use it. If another loop is opened, this information get pushed farther down the stack and will probably not be removed. This is the beginning of a mess.

There are, however, a couple of built-in safeguards. If the loop is within a subroutine, a RETURN will strip off the NEXT information as well. Likewise, if you exit a loop, but use the same loop variable to open a new one, the old "NEXT" information will be removed.

Of course nobody writes programs like this but if you get the ?Out Of Memory Error and FRE(0) indicates plenty of room available, check your loops and GOSUBs.

Stack (Crackle) Pop!

The following SYS's will "crackle" your stack and cause a POP. Apple users will know all about the POP command. It's used to remove one level of subroutine RETURN information. However, this POP simulator removes ALL levels of subroutine returns.

A particularly useful application is within an Input Subroutine. You may have a line that detects a certain character (eg. "@") that exits the Input routine and transfers control to a Command Input routine. Further, the Command routine might test for a special character and exit to a Menu routine.

But, jumping out of subroutines with GOTO can be hazardous (as discussed in the previous section). By using one of these SYS's, the stack is all cleaned up and potential stack overflow is prevented.

```
BASIC 1.0  : sys 50568
BASIC 2.0  : sys 50583
BASIC 4.0  : sys 46610
VIC 20     : sys 50544
C 64       : sys 42352
```

If you just want to POP a single return, code up 7 PLA instructions followed by an RTS and SYS to that. The above sys's are much safer though. Any open loops will interfere with a single POP and may cause adverse side effects.

Universal Reset

Instead of switching off your computer next time you want to clear out memory, etc, try this SYS. It does the same thing without that nasty power interruption.

```
sys peek(65532) + 256 * peek(65533)
```

This works not only on all Commodore machines, but any machine that uses the 6502 as its microprocessor. The 6502 pre-defines addresses 65532 ($fffc) and 65533 ($fffd) as the locations that will contain the address of the machines reset routine.

When you "power up", the 6502 does a "JMP Indirect" to $FFFC. This is known as the Hardware Reset Vector. The HRV will change from machine to machine. Use the above statement for universality; use these when you want to reset from a particular machine:

```
BASIC 1.0  : sys 64824  ($fd38)
BASIC 2.0  : sys 64721  ($fcd1)
BASIC 4.0  : sys 64790  ($fd16)
VIC 20     : sys 64802  ($fd22)
C 64       : sys 64738  ($fce2)
```

Vertical Messages

Want to print a string vertically instead of the same old horizontal way? Easy with the windowing feature of 8032 type and Fat 40 machines.

```
  10 input" ss some string" ; ss$
  20 a$ = ss$ : gosub 1000 : print ss$
  30 goto 10
1000 print : print tab (rnd(0)*40);
5000 print " o ";left$(" qqqqqqqqqqqqqqqqqqqqqqqq ",
     len(a$));" O ";
5010 return
```

The two "Cursor Home" characters in line 10 clear any existing window. SS$ is transferred to A$ to make the subroutine more versatile.

Line 1000 can be removed. It merely TABs a random amount across the screen for the demonstration.

Line 5000 sets a window one column wide and as high as A$ is long. The routine leaves the cursor at the bottom of the window. Upon returning, SS$ is printed and neatly scrolls into place. For one column windows, the scrolling is so fast you'll never notice it.

The special characters in line 5000 are achieved using the sequence of ESCape and RVS. After typing the quote, ESC turns quote mode off, RVS enters reverse field mode, and the letter "o" (Set Top character) is pressed. However, you're still in reverse mode; hit Shift RVS to get out. Same goes for "O" (Set Bottom). Try this program:

```
100 a$ = " = = = = = = = = = = = = = = =
    = = = = = = = = = = = = = = = = = =
    = = = = = = = = = = = = = = = = = = = = "
110 print " S " ;      : rem ↑ 48 = 's
120 a = rnd(0)*48 : i = rnd(0)*80
130 ? tab(i) " o " left$( " qqqqqqqqqqqqqqqqqqqqqqq] " ,a)
    " Or " left$(a$,a);      : rem ↑ 24 down's & 1 right
140 print spc(24 + a/2) " s s " ; : goto 120
```

Pretty useless, eh?

Escape Without Escape

The ESC (Escape) key is found on Commodore machines with business keyboards only. On some earlier machines it does absolutely nothing. On later machines with BASIC 4.0, it serves to cancel "quotes mode"; invoked when an odd number of quote keys (") have been pressed.

However, quotes mode is also invoked when a odd number of quotes are PRINTed to the screen (eg. PRINT CHR$(34);). This can be rather offensive in an Input Subroutine, especially when you don't want to disable the quote key. After the user hits the " key, any cursor keys pressed will be displayed in their reverse field or "programmed" representation. To disable this mode, the user must either hit the ESC key (if there is one) or type another quote and DELete it. What a pain.

The following POKEs will cancel quotes mode. The POKE could be placed after a test for the quote key:

```
get a$
if a$ = chr$(34) then poke. . .
```

. . .or, more simply, executed after every key press:

```
get a$ : poke. . .
```

Here are the POKEs you'll need depending on your machine:

```
BASIC 1.0 : poke 234, 0
BASIC 2.0 : poke 205, 0
BASIC 4.0 : poke 205, 0
VIC 20    : poke 212, 0
C 64      : poke 212, 0
```

If for some reason you want to turn quote mode on, just POKE the respective location with a 1.

Shift Key Detect

This program will detect if the Shift key is depressed.

```
100 if peek(152) then print " shift key down "
101 print " shift key up " : goto 100
```

And YOU say, "so what?". Well, the Shift key has one advantage over other keys in that no character is entered in the keyboard buffer to interfere with your GET and INPUT commands. As part of a piece of software its uses are limited (Superscript uses the Shift key to speed up scrolling through text and output to video). But of even greater significance is during program development. Such a statement could be used to re-direct execution, test variables, change variables, or even alter machine conditions such as toggling Upper/Lower case or STOP key disabled/enabled. For example:

```
2000 print " Press 'S' to Save Record "
2010 x = rn : gosub 10000 : get a$ : if a$ = "" then 2010
2020 if a$ <> " s " then return
2030 record #8, rn
2040 print #8, rn$
2050 . . .

10000 if peek (152) then print x;
10010 if peek (152) then 10010
10020 return
```

Of course any amount of information could be transferred to subroutine 10000. By using a common variable to receive data (ie. "x"), the subroutine remains versatile and can be called from elsewhere in your program.

Line 10010 is a loop that waits for the Shift key to be released. If you hit "Shift" and nothing happens, you've probably pressed it during this line; release and try again.

This merely demonstrates a technique. It could be much more sophisticated than shown. One improvement might

be the addition of cursor position store & restore subroutine calls at the beginning and end of this "pseudo–monitor" (see Jim Butterfield's Cursor Position Recorder this issue). Variables could then be displayed on, say, the top or bottom line of the screen where they won't disturb other screen contents. Then the cursor would be sent back to its previous position to retain normal program appearance.

Once again, the main PEEK will depend on:

```
BASIC 1.0 : peek (516)
BASIC 2.0 : peek (152)
BASIC 4.0 : peek (152)
VIC 20    : peek (653)
C 64      : peek (653)
```

In all cases, if the above PEEK yields a zero, the Shift key is up, otherwise down.

Other possibilities for "PRINT X;" in line 10000:

```
poke 59468,  26–peek(59468)  ;flip case (pet/cbm)
poke   144, 173–peek(144)    ;stop en/disable (4.0)
poke   144,  95–peek(144)    ;stop en/disable (2.0)
poke   537, 269–peek(537)    ;stop en/disable (1.0)
```

SuperPETs With Hard Disks

When using the SuperPET, the language disk is usually in drive 1 of your floppy. From the SuperPET menu, by simply entering the first letter of the language (ie. "a" for APL, "b" for BASIC, etc.), the system goes off to drive 1 to begin loading.

But if you have just added a Commodore 9060 or 9090 Hard Disk, you may have noticed there is no drive 1, only drive 0. Now you must enter language load commands manually. Syntax is:

```
disk/0.APL
```

Of course, "APL" could be any language by choice.

Petunia Users Beware!

Back around 1978/79, an interface for PET/CBMs was released called "The Petunia". It combines an active digital to analog converter and a video interface for connecting to external monitors. The unit works very well on PET/CBMs, but don't use them on your VIC or 64!

The Petunia plugged on at the PET User Port. Video lines are on the top edge and the User Port lines are on the bottom. On the VIC and 64 the User Port lines are still on the bottom edge, but video signal lines are now routed to a connector all their own. Where the video out (pin 2) used to be on the PET User Port, is now +5 volts on the VIC/C64 User Port. If the Petunia "video in" is connected to +5 volts it will fry like a banana!

Now, you ask, "How does a banana fry?"... plug your Petunia into your VIC and you'll find out!

Supermon 64 Correction

In the January Issue of COMPUTE!, Jim Butterfield's Supermon 64 was published with one slight error that will throw a wrench into the works.

At the end of the BASIC listing are three pokes. The middle one should be POKE 45, 235 (not ,232).

Also, Dave Berezowski of Commodore Canada suggests this mod to be included at the beginning of the program:

```
poke 53281, 12
```

The poke sets the background colour to grey which looks much nicer than the blue background from power-up

Dave Berezowski of Commodore Canada

Chart Of Standard Business Procedures

On the next two pages lies our interpretation of an age old comic. We felt it was time for an update.

CHART OF STANDARD BUSINESS PROCEDURES

As Sales requested it

As Production ordered it

As Engineering designed it

As the Lawyers approved it

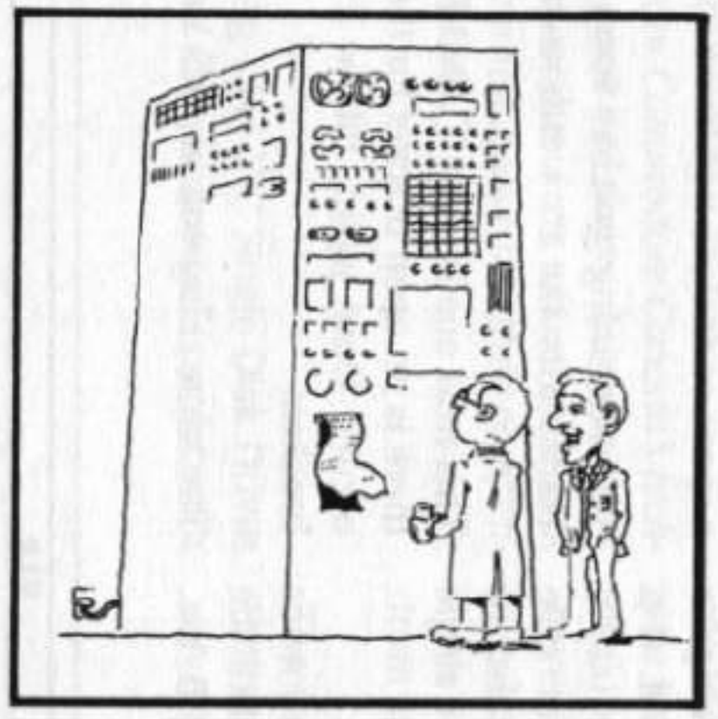
As Management approved it

As Advertising promoted it

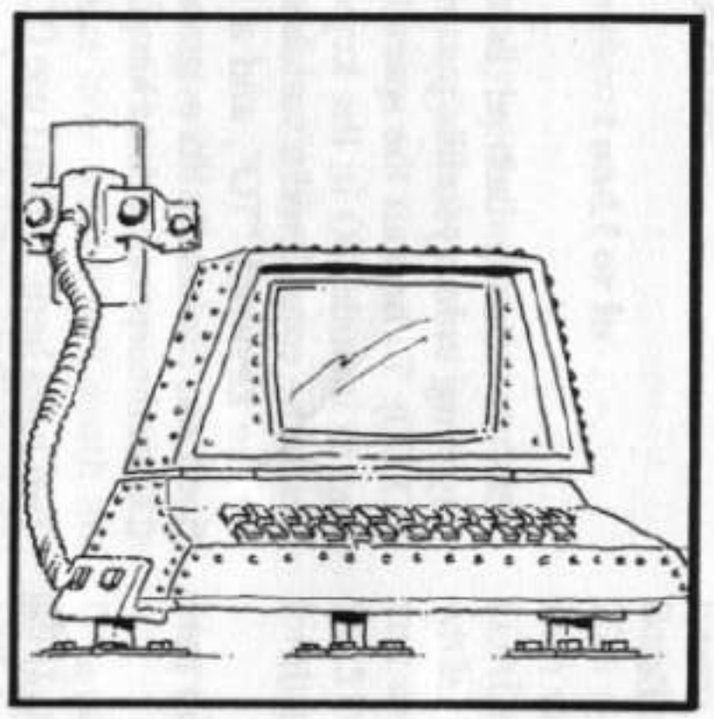
As the Plant installed it

What the Customer wanted

CHART OF STANDARD BUSINESS PROCEDURES

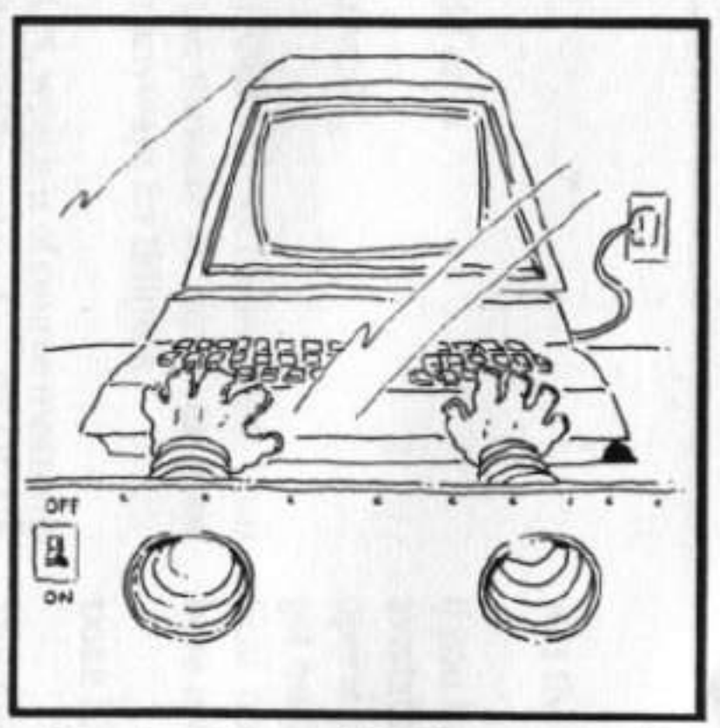


[1] As Sales requested it

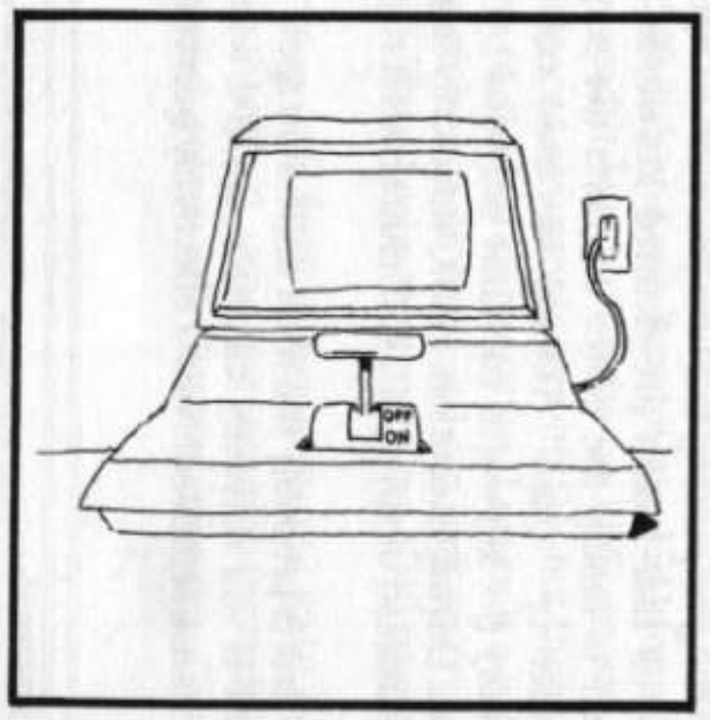


[2] As Production spec'd it

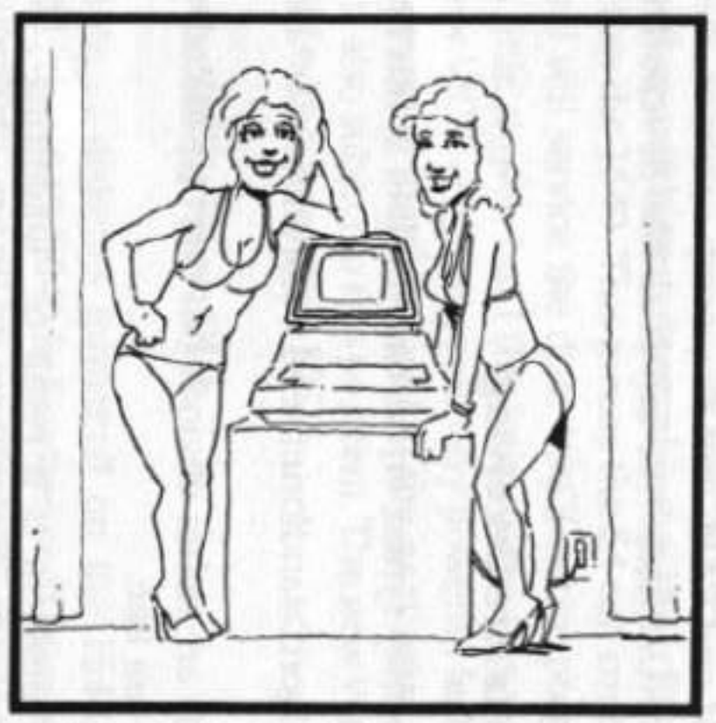
[3] As Engineering designed it



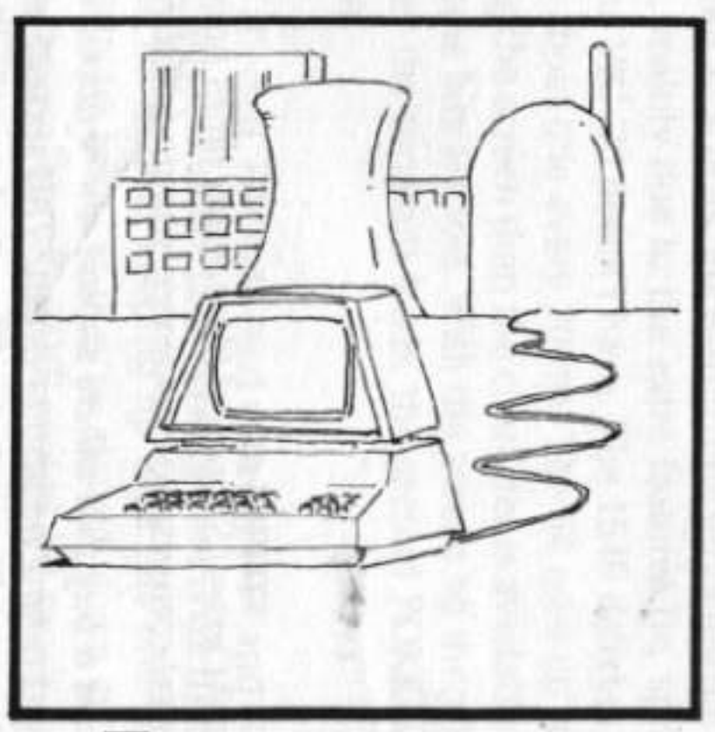
[4] As CSA and FCC approved it



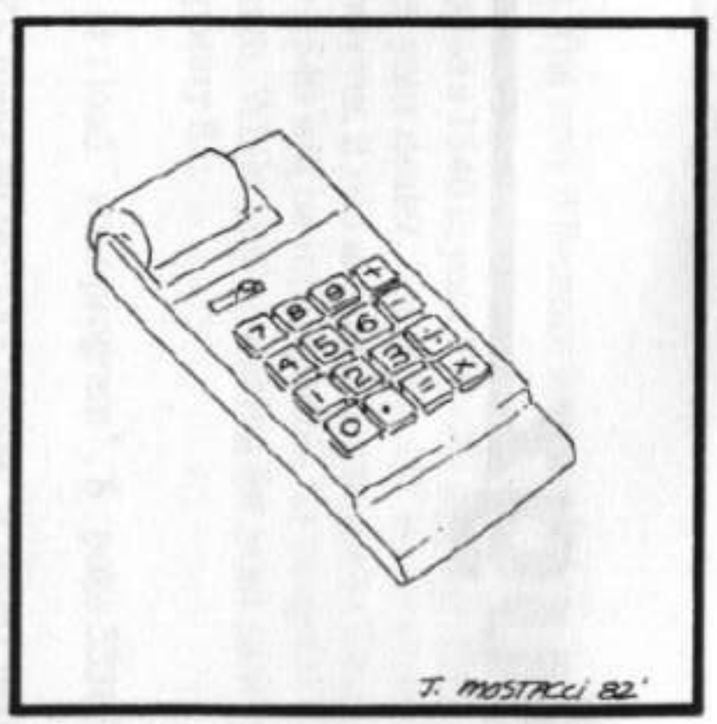
[5] As Management approved it



[6] As Advertising promoted it



[7] As the Plant installed it



[8] What the Customer wanted.

Bugz

Drats! Curses! Shavings! How many times have these words entered conversations between you and YOUR computer?

Found a bug that others should know about, or a potential programming hazard to avoid? This section is an extension of Bits and Pieces designed to deal specifically with bugs in ROM and RAM, and temporary solutions. News of permanent ROM fixes from CBM will also be reported here. Ed.

Looks Are Deceiving!

How may syntax errors can you find in this line:

```
ifgorb>tandforinthend=storun:header"disk",d1,ifn
```

Some are rather obvious but don't be fooled. . . other parts will run fine.

The problem is a result of "tokenizing". When you hit Return to enter a line of text, the BASIC editor begins analyzing or *parsing* the line (from left to right) looking for patterns that match pre–defined keywords in ROM. If a match is made, the editor converts the sequence into a number or "token". Tokens are used not only to save memory (ie. keywords consume only one byte) but also for speed. During execution, BASIC need only interpret a single byte instead of string of 2 to 7 characters long.

The first is probably the most difficult to spot. The sequence G OR B will correctly calculate G OR'd with B, until the spaces are removed! Now it effectively becomes GO RB. Yes, "GO" is actually defined as a keyword that doesn't do anything, unless you like putting spaces in your GOTOs (ie. GO TO). To honour the spaced out "GO TO", Commodore had to include GO in the keyword table. Otherwise "GO" would be considered a variable during execution, followed by the keyword "TO", and a ?Syntax Error would result. Such was the case in the original BASIC 1.0. All BASICs from 2.0 on incorporated this change.

Two rules here: 1. Don't use GO for a variable, and; B. if the variable G precedes the boolean operator OR, follow G with a space or enclose it in brackets.

The remaining gotchas have been discussed in earlier Transactors, but are worth re–mentioning.

This next one is easy. T AND F will read as TAN DF. Since there is no open bracket for the variable DF, ?Syntax Error occurs. Once again, follow T with a space or put it in brackets.

The same thing will happen on FOR and INT until we use

spaces to separate:

. . .t and f or in . . .

is how this part must read to work.

"THEN" is processed next so END will not be detected. Thus no space is needed before the variable D. However, good programmers will insert one anyways for tidiness.

After the "=" comes S TO RUN, which is of course meaningless. For this to work at all, it must be entered as: ST OR UN

Our last syntax error lies in the HEADER command. The "I" delimiter for the *disk ID* is followed by an "F". This tokenizes to "IF" and is definitely out of place. Inserting a space after the I will avoid ?Syntax Error, but "FN" is now tokenized as if it were part of a "DEF FN" sequence. Your disk ID will now be a space and the graphic character ◻, which has a PET ASCII value of 165, the same value as the token for "FN". Solution: A. Don't use F as the first character of a disk ID, and; 2. Don't use a pair of characters that match a keyword. These are FN, GO, IF, ON, OR, TO, or ? (PRINT shorthand).

The statement has one more problem in general that often plagues even the most experienced programmers. At first glance, the condition appears to be:

is **g or b** *greater than* **t and f or in**

Not so! The > symbol is actually operating on the variables B and T. To achieve the above test, brackets must be used to delimit the >:

if (g or b) > (t and f or in) then. . .

You may even require brackets within brackets to ensure correct order of operations. Don't be afraid to use a few extra brackets. . . the time you save later will be well worth the extra bytes!

The remaining info came from Commodore Canada's technical department. I have to hand it to them... Commodore's rare policy of releasing precarious facts about their machines, affords them an admirable image that their competitors all too often choose to forfeit.

Incompatibilityisms: C64

Commodore is now shipping 1541 VIC Single Disk Drives for use with the VIC–20s and Commodore. It's predecessor was the 1540. The only difference is one ROM chip.

If you have added a 1540 to your VIC–20, upgrade ROMs are available from your dealer, but there will be no change in operation. However, if you are planning to get a C64 (or just got one) your 1540 will be incompatible for program loads. Fear not though. While you're waiting for your new chip, here is a temporary fix:

```
poke 53265, 11 : load "your program", 8 : poke 53265, 27
```

The first POKE turns off the 64's screen. It seems that the 6510 uses about 25% of its processing time servicing the VIC II video chip, mainly due to the extra features (ie. sprites, etc.) which the VIC–20 doesn't have. The 1540 delivers bits at the rate of about one every 20 ms. The 64 uses up more time servicing the screen than the character is available for. Therefore, some bits are lost. With the screen off there's no problem. After the program loads, the second POKE turns video back on.

Only input is affected. File data read with INPUT# and GET# will suffer the same horrible fate. Writing data is ok though. The 1540 has no trouble keeping up with bits offered by SAVE and PRINT# because they're slowed down by the screen. With the screen off, SAVE would be much faster (maybe too fast?).

Basicly, (or rather machine languagely) the 1541 ROM is 25% slower than the 1540. This means the 64 can continue with video and still service the disk. You'll notice program LOADs are a little slower, but I think the tradeoff is worthwhile. The new VIC II chip has some very powerful features and we'll be covering them in detail in the next issue. The 1525 Printer is also subject to this problem. The new version for the 64 will be known as the 1525E

More Incompatibilityisms: Disk

One other note. . . diskettes formatted on the 1540, the 1541, the 2031, the 4040 (3040 in Europe) or, if there's any still around, the 2040, can all be read by from any of these models. But don't write "interchangeably". That means if you have a 1540 disk and you insert it in a 1541 (or a 4040, etc.), you can read it but don't write on it! Some say they haven't experienced any problems with this but there have also be reports of diskette clobberation. I make it a habit to have one of each format on hand so I can pick up programs from any drive. Then I copy them onto my 4040 later on.

The same is true with 8050 diskettes on the 8250 drives. One difference though. . . upon first inserting the disk in the drive, your initial access will give an error (eg. Catalog).

Clear the error channel with PRINT DS$ and give the Catalog command again. You should have no further trouble until you insert another disk. But once again, don't write on them. Instead, format a new 8250 diskette and use COPY to transfer 8050 Program and Sequential files across. If you have Relative files, you'll need to use a utility to make the transfer because REL files are formatted differently on the 8250. (Jim Butterfield's COPY–ALL will do it)

1540/41 Command Change

All Commodore disk units support the Memory–Write command. This command works like a "disk POKE" for writing data into DOS memory. On 2040, 4040, 3040, 2031, 8050, 8250, 9060 and 9090 drives the syntax is:" m–w: " (followed by an address and data, see your disk manual for details). Syntax on 1540 and 1541 drives is:" m–w " (ie. without the colon).

This command is not to be used haphazardly. Memory–Write deposits data in DOS memory that may or may not send it into never–neverland.

VIC–20 Printer Output Bug

If a program is LISTed to the 1525 printer from the VIC–20 immediately after a SAVE to tape, the 1525 will drop characters. For example:

"beige tint" might become "big ti . . .hmm, poor choice

Well, you get the point. The fix? Simple. After the SAVE, type the following:

VERIFY <Return> <RUN/STOP>

It seems that activating the VERIFY command clears the adverse condition created by SAVE. RUN/STOP aborts the VERIFY and you can now send unbugulated listings to your printer.

Alternately, you could LIST your program before SAVE. After the SAVE, a LOAD will untangle the output routines like VERIFY.

No Interlace On VIC II Chips

Some televisions on the market have what's called "interlaced CRT scan". This means that the video beam scans all the *even* rasters during one sweep, then goes back and scans all the *odd* rasters on the next sweep. Other TVs simply scan consecutive rasters.

VIC–20 video chips have a feature called "interlaced mode". To activate it:

poke 36864, 133
poke 36864, 5 de–activates it

If your picture appears to "flutter", try the above POKE. It may or may not help. Note that game cartridges from VIC–1910 up, with one exception, allow you to toggle this feature by hitting the F7 key before the game is started. The exception is "Gorf" (VIC–1923). With this cartridge, push the joystick up instead of hitting F7.

Back to the point. . . The Commodore 64 uses a new video chip called the VIC II. This chip doesn't have the interlace mode feature. Although this is not a bug, it was included in this section because it might look like one. If it happens to you, I'm afraid you're stuck. However, the 64 has pretty good video output. Chances are you won't notice it even if you have a TV with interlaced scanning.

Zenith TV Mod

On some recent Model 3 Zenith TVs there is a problem with the vertical hold synchronization. This not only affects the 20 and the 64, but virtually any device used locally to drive it (ie. other micros, video games, VCRs, etc.). The picture will, once again, appear to "flutter" because Zenith factory sets the unit to receive its vertical sync (or interlace) signal "off air". (the signal is mixed in by the station — you may have heard the buzzletters "V–I–P" used to promote this product)

The POKE discussed in the previous segment can be used to fix it, but only for VIC 20s. However, Zenith offers this more permanent fix:

Inside the Model 3 lies a yellow wire on connector 2H of module 9–152. Disconnecting this wire will force the set to generate its own internal sync signal. This might seem simple enough, but have a dealer or qualified technician do it for you. Your warranty won't be voided, nor will you notice any change during regular TV viewing (most TVs like Sony don't even use off–air sync).

Note: There is a white wire connected next to the yellow wire which should NOT be disconnected. This problem has been observed on other Zeniths and some RCAs but no specific model numbers or fixes are available at this time.

Commodore 64 Bugs Update

Here is a list of all known 64 bugs to date:

1. TAB and SPC

The PRINT# command cannot be followed directly by a TAB or SPC operator. To get around this, simply precede TAB or SPC with two quotes or a "literal null string". Eg:

```
open 4, 4
print#4, "" tab(10) " some string "
```

2. Prompt Suppress After CONT

If a program is interrupted with the RUN/STOP key, and CONT is entered to resume execution, the prompt messages generated by the operating system will no longer be suppressed. For example, if you have CONTinued a program and a dynamic LOAD occurs, ie:

```
100 load " next module ",8)
```

the prompt, "searching for next module" will be displayed on your screen. This one is really no big deal so there's no fix, although a POKE to location 19 before using CONT might do the trick.

3. Screen Editor Crash

This one was found pretty early and you may have already heard about it. Let's say you're on the 23rd, 24th, or 25th line of the screen and you type a line that's longer than 80 character but less than 120. If you now begin deleting characters, upon deleting the 80th character (DELete from column 1 of the 3rd line *around* to column 40 of the 2nd line), your machine will appear to hang.

Apparently, upon writing a space to this location, the 64 incorrectly writes information outside of the colour table. This info actually gets written to CIA 1 which is just above the colour table ($DC00). If a certain bit is set, CIA 1 will invoke an auto LOAD/RUN (as if you hit Shift RUN/STOP). This bit will be set or unset depending on the colour of your cursor. At this point, your keyboard will seem to be disabled.

Fear not! Here is the fix (thanks to Dave Foster of Richmond Hill, Ont). If the "9" and the "N" keys are depressed together and then released, the prompt "PRESS PLAY ON TAPE" will appear. Do so and the screen will display "OK" and go blank. Now press the RUN/STOP key and you will regain control of your 64 with no apparent ill effects. (Note for disk users with no Datasette: connecting the cassette port pins 1 and 6, the outside pins, together will have the same effect as pressing PLAY)

To avoid this potential situation altogether, simply change the cursor colour to white (usually best), purple, green, orange, brown, grey2, or bright green.

New Kernal ROM For 64

Commodore has been installing new Kernal ROMs in their latest production 64s. Dubbed the "Kernal 2", it fixes bug #1 from above (and maybe #2 but not #3) and also incorporates some changes. However, it's already been discontinued in lieu of a "Kernal 3" ROM coming soon. Rumour has it that Kernal 2 machines will be updated to Kernal 3, but Kernal 1 machines will have to wait. The reason for this is a change in Kernal 2 that was made too late for it to be effective. Kernal 3 will revert back to the original method used in Kernal 1.

The change in point is this. A "Screen Clear" with Kernal 1 resulted only in the screen being written with space characters. The colour table was left untouched which means any subsequent POKE to the screen would produce a character in the previous colour assigned to the location being POKEd.

Kernal 2 works the same way as the VIC 20. Clearing the screen causes the entire contents of the colour table to be written with the same value as the background colour. Now, a POKE anywhere to the screen will produce a character that is the same colour as the background. Thus it will appear to be "not there".

Had Commodore released the original Kernal 1 C64 with the VIC–20 clear screen procedure, there would be no need for a Kernal 3. Kernal 2 renders several 64 programs already in circulation inoperative after a clear screen (WordPro 3+ 64 for one). This may be a blessing in disguise though, since new bugs have been discovered since the release of Kernal 2.

Other changes in Kernal 2/3 include a slightly different operation of the Commodore Logo key when LOADing from tape. In Kernal 1, a tape LOAD would cause the tape deck to find the program and wait indefinitely for the Logo key to be pressed before proceeding. This gives you the option of aborting the LOAD by hitting RUN/STOP. With Kernal 2/3, the 64 waits 10 seconds for you to hit the Logo key once the program is found. After 10 seconds, LOAD proceeds as if you did hit Logo.

To check for Kernal 2, enter the following line:

print"■" : if peek(55296) = peek(53281)+240 then print"Kernal 2"

Tune in to this section next issue for more news on Kernal 3. Additional changes are expected.

Best Monitor Picture From VIC/64

Here are the pin designations for the 5 pin Video/Audio connector on VIC 20s and C64s. The colours are those of the Radio Shack 5 pin European plug to 4 phono jack cable (Part# 42-2394).

	VIC 20	C64
1 Red	+5V @ 10ma.	Luminance
2 No Lead	Ground	Ground
3 Grey	Audeo Out	Audeo Out
4 Black	Video Low	Video.
5 White	Video Hi	SID Audio In

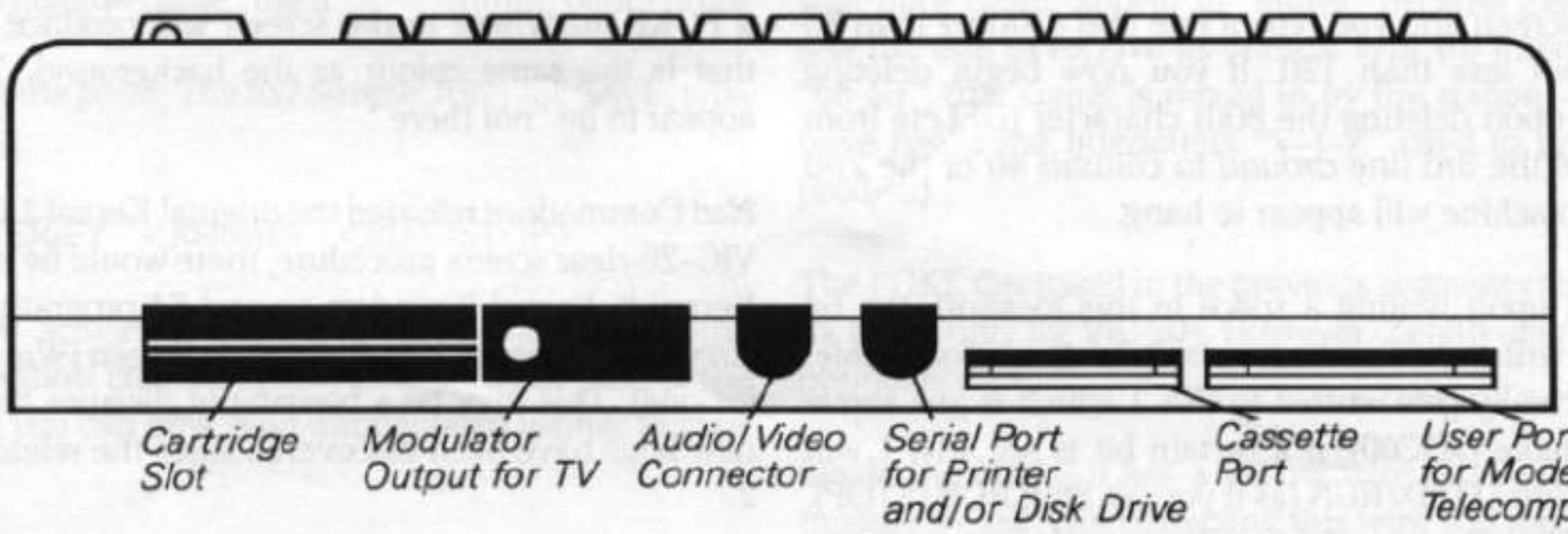
If you have a VIC 20 and a B&W or colour monitor, try connecting pin 4 (Video Low) or pin 5 (Video Hi) to your monitor input. Whichever gives you the best picture will obviously be the one to use.

Commodore 64 users with video monitors have a couple of options available. For colour monitors, get yourself a phono "Y" adapter from Radio Shack. To this connect Luminance (pin 1, Red) AND Video (pin 4, Black) and plug the Y-adapter into your monitor. You should get a much sharper picture than with just Video alone.

For 64s with B&W monitors, connect just Luminance (pin 1, Red) to the monitor input. Video seems to give a "grainy" picture but Luminance comes through nice 'n' clear!

DON'T however try either of these with your VIC 20. You'll be connecting +5 volts to your monitor and you could be in for a spark show!

One last note. . . new 64s are coming soon that will have an 8 pin Video/Audio jack. No details yet on the extra 3 pins, but we'll keep you posted.



Transbloopors

This section, although hopefully not regular, will cover publication errors and/or improvements for previous Transactors. If you find a significant mistake in an article or program, please let us know so we can pass it on. Ed.

Indistinguishab 'l's

First off, I'd like to mention that we're working on the problem of distinguishing lower case L's from the number 1 in program listings, etc. For now though, a lower case L will have a "flat" top, and the top of a 1 is cut at a slight angle. Usually it's not hard to tell which is which merely by where they occur. But I admit, there are some dubious cases. Please bear with us until next issue.

Bits and Pieces **Vol4 Iss1 Pg6.**

As it turns out, SuperPET Cobol is not free to existing SuperPET owners. I'm not clear whether it's included with new SuperPET purchases, but ask your dealer for details.

Disk Un–Assembler **Vol4 Iss1 Pg19.**

Paul Blair of Holder, Australia writes in with these corrections and additions:

"The Paul Higginbottom unassembler is very good indeed, despite the minor problems with the listing. The most important fixes are:

Line 120 sets AL = 251; Line 121 sets A1 = 0.

This problem stems from our L's vs. 1's problem.

Line 809 should read, if ds then. . .(not dd)

Line 841 should read, if mn$(mn)<>"" (ie. null string)

The first pass (to get the file length) is needlessly slow – certainly not in keeping with the rest of the program. To overcome this, I've added a little machine code to speed it up. The main changes are:

1. Extra DATA Statements

```
116 data 169,   0, 162,   4, 149,  95, 202,  16, 251,
         169, 160, 133,  94, 162,   2,  32, 198, 255
117 data 230,  98, 208,  10, 230,  97, 208,   6, 230,
          96, 208,   2, 230,  95,  32, 228, 255, 165
118 data 150, 240, 235,  32, 204, 255, 198,  94,   6,
          98,  38,  97,  38,  96,  38,  95,  16, 244,  96
```

Line 110: change 654 to 709.

The m/c is not original – it appears in several other utilities and seemed appropriate for this one.

2. BASIC Changes

Add ": gosub 9700" to the end of line 3010

```
Add 9700 open 15, 8, 15 : a = 0 : poke 1, 143 : poke 2, 2 :
         a = a + usr(0) : e = s + a : close 2 : close 15 : return
```

Add 3011 print "(";e–s+1;"bytes)"

Delete line 3021

Filing It **Vol4 Iss2 Pg31.**

Two "+" symbols went astray from Jim's article due to a typesetter translate oversight. On page 31, left column, there are two lines containing the sequence, "3*45*6", one around 13 lines from the top, the other about 33 down. Insert the plus signs between the 4 and the 5. (How many thought it should be "3+45-6", besides you Jim)

Also, page 31, right column, Line 130 should read:

130 IF ST=0 GOTO 110 (not 1100)

SuperPET Terminal Program **Vol4 Iss1 Pg32.**

The "tables that follow", didn't follow. Here they are:

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	–	–	1	0	0	0	0
PROGRAM RESET	–	–	–	–	–	0	–	–

Status Register

VIC Modem Driver **Vol4 Iss1 Pg46.**

Move the "next" in line 300 to a new line (ie. 305 next)

Also, for some reason the "return" on line 310 causes a ?Return Without Gosub Error. We don't know why, it just does (any ideas someone?). To get it working:

Change 100 to: 100 goto 200
Change 310 to: 310 goto 110

C64 Preliminary Review **Vol4 Iss2 Pg48.**

The article mentions two Digital to Analog converters in the 6526. These are actually in the SID chip, not the CIA.

CONTROL REGISTER

7	6	5	4	3	2	1	0

STOP BITS

0 = 1 Stop Bit
1 = 2 Stop Bits
1 Stop Bit if Word Length = 8 Bits and Parity*
1½ Stop Bits if Word Length = 5 Bits and No Parity.

WORD LENGTH

BIT 6	BIT 5	DATA WORD LENGTH
0	0	8
0	1	7
1	0	6
1	1	5

RECEIVER CLOCK SOURCE

0 = External Receiver Clock
1 = Baud Rate Generator

BAUD RATE GENERATOR

0	0	0	0	16x EXTERNAL CLOCK
0	0	0	1	50 BAUD
0	0	1	0	75
0	0	1	1	109.92
0	1	0	0	134.58
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	1800
1	0	1	0	2400
1	0	1	1	3600
1	1	0	0	4800
1	1	0	1	7200
1	1	1	0	9600
1	1	1	1	19,200

*This allows for 9-bit transmission (8 data bits plus parity).

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	0	0
PROGRAM RESET	–	–	–	–	–	–	–	–

Control Register

COMMAND REGISTER

7	6	5	4	3	2	1	0

PARITY CHECK CONTROLS

BIT 7	BIT 6	BIT 5	OPERATION
–	–	0	Parity Disabled - No Parity Bit Generated - No Parity Bit Received
0	0	1	Odd Parity Receiver and Transmitter
0	1	1	Even Parity Receiver and Transmitter
1	0	1	Mark Parity Bit Transmitted, Parity Check Disabled
1	1	1	Space Parity Bit Transmitted, Parity Check Disabled

NORMAL/ECHO MODE FOR RECEIVER

0 = Normal
1 = Echo

DATA TERMINAL READY

0 = Disable Receiver/Transmitter (DTR high)
1 = Enable Receiver/Transmitter (DTR low)

RECEIVER INTERRUPT ENABLE

0 = IRQ Interrupt Enabled from Bit 0 of Status Register
1 = IRQ Interrupt Disabled

TRANSMITTER CONTROLS

BIT 3	BIT 2	TRANSMIT INTERRUPT	RTS LEVEL	OTHER
0	0	Disabled	High	–
0	1	Enabled	Low	–
1	0	Disabled	Low	–
1	1	Disabled	Low	Transmit BRK

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	0	0	0	0	0	1	0
PROGRAM RESET	–	–	–	0	0	0	1	0

Command Register

The WordPro Book Of Tricks

Donna Green
Commodore Canada

Focus on Deleting and Erasing Functions

Because many of the users of WordPro could be first time users of any word processing system, we will address, in this issue, some of the very basic functions in text editing, mainly erasing and deleting. There actually is a difference between the two: "deleting" functions usually result in the text "wrapping" on the screen, while "erasing" functions leave behind a space of some kind.

Most WordPro users will quickly discover that there are often several ways to do the same thing. In this issue we will have a closer look at the various ways of deleting. Perhaps there will be some new shortcut you haven't yet tried.

Command Summary for "Deleting" Functions

1. Control + d + w + <Return>

2. Control + d + s + <Return>

3. Control + INST/DEL(deletes an entire line)

4. Spacebar – Pressing the spacebar in the middle of text will replace any typed characters with spaces. This is a necessary feature because it allows for corrections to be made by "striking over".

5. Return Key – Used only at end of paragraphs and short lines such as headings. When pressed in the middle of a line of typing the rest of that line will be deleted. This is a quick and easy way to revise the end of a paragraph, and a very logical function when you realize that a carriage return is always the last key pressed at the end of a paragraph.

   Note: A Shifted Return will return the cursor to the line below, to column one, and will not erase the rest of the line!

1. Deleting Characters, One at a Time

(a) Cursor Position – Move one character to the right of character to be deleted.

(b) Command – See Step 1. above. Press the INST/DEL key once for every character to be deleted.

(c) Caution – Watch the screen to avoid deleting more text than required. If text inadvertently "wraps" up to the line above insert enough spaces to force it to "wrap" down to the proper line below. This would probably only happen if many characters were being deleted at that time, in which case probably the method below would be better to follow for deleting several words or sentences.

Note: A "Space" is considered a "Character" and may be inserted or deleted like a regular letter or number!

2. Deleting Complete Words and Sentences

(a) Cursor Position – Move to beginning of word/sentence to be deleted. Cursor may be on the first character of word to be deleted or on the space preceding the word to be deleted.

(b) Command To Delete a Word – See Step 1. above. Press:

"Control" + "d" + "w" then <Return>

(c) Command To Delete a Sentence – See Step 2. above. Press:

"Control" + "d" + "s" then <Return>

Note: The letters "w" or "s" may be pressed more than one time, and even combined if several words or sentences are to be deleted at once, e.g.

Control + d + w + s + s + w + w + then <Return>

This would delete one word, two sentences, and two more words of the next sentence.

3. Deleting Paragraphs – One Line at a Time

This method is rather slow if large amounts of text are to be deleted, but ideal if only a few lines are involved.

(a) Cursor Position – Anywhere on the first line to be deleted.

(b) Command – See Step 3. above. Press Control then the INST/DEL once for every line to be deleted.

(Don't forget that the "C" for "Control" mode is still highlighted on the status line, so press the control key a second time to turn off "Control", before continuing with any editing.)

4. Deleting Paragraphs – By Setting a Range

This method is quicker if large amounts of text are to be deleted.

(a) Cursor Position – Anywhere on the first line to be deleted.

(b) Command – First define the paragraph, or amount of text to be deleted, by setting a "range". To "set a range", follow the steps below:

Press: Control + r(range)	entire line will be highlighted
Press: "cursor down" key	next line will be highlighted
Press: "cursor down" key again	until entire paragraph is highlighted
Press: Return key	to freeze the paragraph in memory
Press: Control + e(erase) + l(lines in range)	entire para. will delete from screen

5. Deleting Paragraphs – Leaving a Blank Space

(a) Cursor Position – Anywhere on the first line to be deleted.

(b) Command – See Step 4. above. The Return key will erase a line of text and leave a 'L' on the screen where the Return is pressed.

6. Deleting An Indented Area

Example – A document has been typed with all paragraphs indented 5 spaces (example below). Now it has been decided to remove all indented areas.

WordPro 4 Plus offers an exceptional text editing including every entering, editing, memory, and printing feature considered important to sophisticated word processing.←

Method One

(a) Cursor Position – On the first letter in the sentence, on the letter "W" in WordPro, in this example

(b) Command –Press the "INST/DEL" key once for every "space" to be removed, until the letter "W" is at column one.

Method T wo

(a) Cursor Position – On the space preceding the first letter in the sentence and type a period – example:

.WordPro 4 Plus offers an exceptional text editing

including every entering, editing, memory, and printing feature considered important to sophisticated word processing.←

(b) Cursor Re–Position – Move cursor back to column one on the same line – and give the command to delete a sentence. (The system recognizes a "sentence" as everything up to a "period"!)

(c) Command –Press "Control" + "d" + "s" + <Return>

The text will move to the left. Individual spaces may be deleted simply by pressing the INST/DEL key.

7. Deleting Indented Areas (Moving Columns Left)

Example – A list of items has been typed all beginning at column 15 (example below). Now it has been decided to move the list of items over to column one.

house←
boat←
car←
cottage←

(a) Cursor Position – On the first letter of the word (the letter "h" in house, in this example)

(b) Command –Press the "INST/DEL" key once for every "space" to be removed, until the letter "W" is at column one.

(c) Repeat – For the remaining items in the column.

8. Deleting An Indent Faster

(a) Cursor Position – On the space preceding the first letter in the sentence and type a period – example:

.house←
.boat←
.car←
.cottage←

(b) Cursor Re–Position – Move cursor back to column one on the first line – and give the command to delete a sentence. (The system recognizes a "sentence" as everything up to a "period"!)

(c) Command –Press "Control" + "d" + "s" + <Return>

(d) Repeat – For the remaining items in the column.

9. Joining Two Paragraphs Together

Using the method described above where an "end of sentence" or a period is typed at the beginning of a sentence, the chore of joining two paragraphs together becomes a simple task.

Step 1 – Insert a space and type a period before the first character in the second paragraph, in this example, the "I" in the word "It" in paragraph 2.

Example:

WordPro 4 Plus offers an exceptional text editing, document storage, and typewriter quality printing capability to any business whose needs would benefit from the increased productivity inherent to word processing.←
←
.It includes every entering, editing, memory and printing feature considered important to sophisticated word processing.←

Step 2 – Cursor up one line and delete the blank line (Control + INST/DEL)

Step 3 – Cursor up again, and position cursor one space after the period (right on the carriage return symbol (the '←' character).

Step 4 – Used the spacebar to remove the carriage return symbol, then give command to delete a sentence. Press Control + d(delete) + s(sentence) and Return. – The two sentences now become one paragraph!

10. Deleting Files From The Diskette

To the relief of many first time word processing operators, deleting WordPro files accidently from the disk is impossible. Below is the command:

Control + .(period) + s(scratch) + [file name] + <Return>

Recall the Directory to see that the name of the file no longer exists (Control + 0 or 1 and return).

That's all for now! More next time.

Bulletin Board Numbers

"Bulletin Boarding" should be classified as another national pastime. Below is a list of Bulletin Board phone numbers in the Toronto area alone. We would like to make the list more international, so if you're running a board or know the numbers in your area, please send them in. From time to time we'll publish the list when enough additions or changes warrant doing so.

This first list was compiled by Richard Bradley, Sysop of the NORTEC BBS.

Bulletin Board	Operating Hrs.	Number	New Number
NORTEC BBS *	24 Hours	487-2593	
PSI-WordPro *	7pm-9am	624-5431	
TPUG *	7:30pm-9am	223-2625	
RTC BBS *	5pm-9am	884-6198	
Atari Info-System	24 Hours	622-2462	
Apple-Can	24 Hours	447-8458	
ETI Bull BBS	5pm-9am	423-3265	
Infoport	24 Hours	278-3267	
Toronto Net-Works 1	24 Hours	445-6696	
Toronto Net-Works 2	24 Hours		
THUG	7pm-7am	231-4174	
RCPM 1	24 Hours	231-9538	
RCPM 2	24 Hours	231-1262	
IBMPC BBS	24 Hours	499-7023	
OSBOARD		626-8066	

(* = Commodore Based System)

Area Code is 416 for all numbers

The numbers for THUG and OSBOARD may have changed by the time you read this. Check with another board for an update before trying these two numbers.

The Toronto Net-Works 2 number has been omitted. To get this number requires a call to Toronto Net-Works 1.

The MANAGER Column

John Stoveken
Milton, Ont.

MANAGER Users Unite!

Just to let you know that you are not alone, there are approximately 6000 MANAGER users throughout the world. The software has been in many different environments, and here are a few that we have been informed of.

The **World Water Ski Championships** have been scored and reported with The MANAGER for the last two years. Each national team had their event points totaled and then reported with the standard package.

The **United States Navy** has been using The MANAGER in their shipyards to track the reworking of their ships in dry dock. The Philadelphia shipyard was the first to implement the software.

Speaking of navies, the **Royal Navy** had a PET on the aircraft carrier HERMES that was used to keep track of Argentine prisoners during the Fauklands crisis.

The **Rothman Cup Horse Jumping Championships** were scored and all stall, feed and veterinarian fees were billed using The MANAGER last summer.

Educational Software for the PET/CBM that is available in Canada, U.S.A., W. Germany, Israel and in the U.K. has all been recorded on The MANAGER.

The **Cleveland Police Department** tracks all their violations and arrests with The MANAGER.

Where else is The MANAGER used? We know of many applications in churches, manufacturing scheduling and inventories (do you know that paint ingredients are priced to hundredths of a cent), administration and laboratories. If you have an interesting application that you are using The MANAGER for we would like to publish it here.

ARITHMETIC Goodies. . .

When calculating averages it is important to determine whether a number has been entered into the field or not. This is because the math will sum a blank field as zero and this can shift the average to zero. The solution to this dilemma is to use a register as a counter like this. . .

```
F5/F5 TO R1      ;if F5 is 0 or blank then R1 = 0
                 ;          otherwise R1 = 1
R1 + R3 TO R3    ;sum the number of entries
F5 + R2 TO R2    ;accumulate the value of F5
R2/R3 TO 2D1     ;display the average with 2 decimal digits
```

In some applications a fixed numeric rate may be associated with a file, or a math operation may be optional depending on some condition. In an invoicing scheme we have a case

where individual item prices must be summed (1), then a hourly rate may be applied to the invoice (2), and then a fixed tax applied depending on the users status (3).

We can then imagine a data file where fields (data types) 5, 7 and 9 are item prices, field 10 is the number of hours charged, and field 11 a tax exemption number. Now, using The MANAGER math:

```
F5+F6+F7 TO R1          ;sum the line items(1)
27.5 TO R2              ;place the hourly rate in a register
                        ;note that it is not carried
                        ;in the data record
F10 * R2 + R1 TO R1     ;calculate the hourly charge
                        ;and add it to the total(2)
F11/F11 -1 TO R3        ;divide the exemption number,
                        the result
                        ;is -1 if there is no number and it
                        ;is  0 if the number has been
                        entered
R3 * .07 * R1 TO R4     ;places 0 in R4 if there is a tax
                        ;exemption number, or places the
                        ;- 7% in R4 tax if taxable(3)
0-R4 TO 2D1             ;display the tax
R1-R4 TO 2D2            ;display the total bill
```

REPORT GENERATE Goodies

The report package of The MANAGER, when integrated with the sort option gives you access to a very powerful "control break" report package. This allows you to group your data logically and to produce logically separate reports from one data file in one step.

For example, if we have an Accounts Receivable file that has one entry for each receipt that contains the customer i.d., account classification, amount owed and invoice date, we can report the data in several different formats. If we sort the file by date and report only the non-zero amount owed, we can produce an aged receivables listing instantly.

A control break report will allow us to produce individual invoice sheets for each customer in the file in one pass. All that is required is to first sort the file with the customer i.d. as the primary key, the account class as the second key and the invoice date as the third. When we report using this pointer file we will have then grouped all the records first by customer, then bill type (capital, service, consulting) and finally by date.

The print conditions should then specify that the -1- break be on the customer with paging, and that the -2- level break be by account type with linefeeds. The resultant report will then have each customer listed on a separate sheet of paper and this can be sent directly to him to inspire prompt payment.

Back To BASIC

The screen dump routines (\d) and special print routines (\p) in The MANAGER refer to two memory locations to determine the printer type and it's IEEE device number. These two locations are:

```
address 22527: 12 = CBM matrix printer
               14 = ASCII printer

address 897  :  4 = default printer device number
                5 or other optional
```

While doing control break reports, there are times when you may want to print just subtotals and totals, not the individual line items. This can be done by "rem-ing" line 1910 in the program "generate2". Before you remove the line do make sure that it reads:

```
1910\p,l$;cl$;
```

Forgotten Backslashes

Last month we forgot to mention one "backslash" command that is one of the BASIC extensions in The MANAGER. The missing command is the machine code save command. It is used to save any region of memory to disk (the screen, for example) for later recall with a \l;"name" command. The syntax of the save is:

```
\s,"name","start","end"
```

with the start and end addresses given in hexadecimal. For example, to save a MANAGER screen we would use the command:

```
\s,"@0:name.scr","6500","7dff"
```

. . .since screen layouts are stored in memory from $6500 to $7dff.

That's all for now. See you here next issue.

Jim Butterfield

The Four–Poke Screen Marker

Jim Butterfield
Toronto, Ont.

Have you ever wanted to pick up a fixed place on the screen, and write something there?

You can do it with cursor movements, of course. Home the cursor and count down and across a given number of locations. That's often the best way . . . it works without change on all Commodore machines .. PET, CBM, VIC, and 64. Provided, of course, that you don't try to go to column 30 on a VIC.

Sometimes it's very useful to be able to mark your place on the screen – something like a bookmark – go somewhere else, and then come back to pick up where you left off. I call this sort of thing a "marker" program; it keeps your place for you.

How does the computer know where things are kept on the screen? It uses four locations:

PNT – two bytes pointing to the memory address where the screen line starts;
PNTR – one byte indicating where the cursor is on that line;
TBLX – one byte giving the line number. This is used only when you go to the next line, where TBLX will be recalculated.

Now – if we save these four bytes, we can put them back at any time and the screen will immediately pick up its former position. We've got a marker!

There's a fifth possible location that might or might not be useful. It's called LNMX and tells you how big the line is – 40 or 80 columns or whatever. Usually, you won't need it; but we'll note that it's there.

Let's look at where these control bytes are kept in the various Commodore machines:

	Original ROM PET	Other PET/CBM	VIC/C64
PNT	224 & 225	196 & 197	209 & 210
PNTR	226	198	211
TBLX	245	216	214
LNMX	242	213	213

So how do we use all this? Let's try a simple program to print prime numbers. We will print the number we are working on at the top of the screen, and each prime, as we find it, in a continuous stream.

The first line depends on your type of system. If you have an antique PET, complete with tiny keyboard and original ROM set, you'll type:

```
100 s0=224 : s1=225 : s2=226 : s3=245
```

If you have a more recent PET or CBM, you'll want:

```
100 s0=196 : s1=197 : s2=198 : s3=216
```

Finally, the Vic or Commodore 64 will need:

```
100 s0=209 : s1=210 : s2=211 : s3=214
```

On to our main program:

```
110  print chr$(147)
120  for j=5 to 1999 step 2
       (we'll start our list of primes at 5)

130  m0=peek(s0)
140  m1=peek(s1)
150  m2=peek(s2)
160  m3=peek(s3)
       (mark our place)

170  print chr$(19);str$(j);"          ";
       (print the number under test)

180  poke s0,m0
190  poke s1,m1
200  poke s2,m2
210  poke s3,m3
       (restore to the marker)

220  for k=3 to sqr(j)+.5 step 2
230  q=j/k
240  if q=int(q) goto 300
250  next k
260  print j;
300  next j
```

A few notes: CHR$(147) clear the screen and CHR$(19) homes the cursor. We don't need to use the fifth value, LNMX, because we don't change it. (notice the semicolon at the end of line 170? That means that we only print a partial line and then go back where we came from). If we did extensive printing at line 170 – or dropped the semicolon – the program would start to get confused between 40-column and 80-column lines and might do the wrong thing.

This isn't an optimal prime number generator, but it works .. and we can see the two screen areas moving at the same time.

Summary

Using four (maybe five) PEEKs and POKEs, we can easily generate a "marker" that allows us to move somewhere else on the screen and then return.

We could keep several markers, of course. And it's our option: we don't have to return if we don't want to.

Determining Screen Size

Chris Siebenmann
Toronto, Ont.

The program below shows my approach to the problem of determining screen size for such things as games and utility loaders (eg. a loader for a word processor). IT IS NOT A COMPLETE PROGRAM. It is a subroutine with three entry points. To use it you tack it on the end of your program and add the appropriate JSR.

How To Use It

First, add it to your program. As written it is completely relocatable, so you can put it anywhere. After a JSR to the appropriate entry point, the screen size is returned in the .Y register (or 128), and a completion code is returned in .X. Note that the value in .X is non-zero when there has been an error and, on exit, the status is set to the value in .X so you can test it immediately by the code:

```
        JSR  entry
        BNE  error
        STY  slen    ;screen len
```

The error code that is returned in .X has the following values. If it is 1 then the program hasn't been able to find the character which it put on the screen. This shouldn't occur, but it probably means that the screen pointer is pointing to the wrong area in memory (if you've used DETENT). If .X is 2 then the program has found the wrong character in memory. The probable cause is from using the wrong entry point (VICENT on a PET or vice versa).

The entry point that you will use depends on the type of equipment you want the program to run on. If your program will never be loaded on a PET, then you need only use VICENT, and vice versa. I usually use both, as this allows me to test for either type of equipment. Thus I use the code:

```
        JSR   petent
        BNE   onvic
        STY   slen
        LDA   #0
        STA   petvic
        JMP   cont
onvic   JSR   vicent
        BNE   error
        STY   slen
        LDA   #1
        STA   petvic
cont    . . .
```

Where "petvic" is a location which tells me which computer I'm on.

How It Works

This is one of three techniques I know of for determining screen size. The first is a ROM PEEK, chancy and must be changed every time Commodore issues a new ROM set. Another is screen overlap. Basically this relies on the fact that there is an extra copy of the screen (1K above $8000 on 40 columns) and therefore will not work on the VIC. This method is universal (as long as the start of the screen is known, hence the VIC entry point) and will even work on 8032's that have been modified by a 40 column shifter utility. It works by printing a character one line down (using cursor down) and then trying to find it in screen memory.

Entering This Program

To enter this program you can either type in the assembly listing (PS. it's in PAL format) or the hex dump (listing two). The hex dump can be put anywhere and all you have to do is remember the appropriate entry address(s). When Start equals the start of the program:

```
PETENT  ¡ Start
VICENT  ¡ Start ‡ 4
DETENT  ¡ Start ‡ 9
```

In summary, I hope this routine helps you as much as it has helped me.

Listing One: Source Code.

```
                ;Determine screen size for PET and VIC
                ; – uses all registers. Clears window on
                ;80 column but will not affect 40 column shifter
                ;.X holds completion code
                ;           0 ¡ everything OK
                ;           1 ¡ can't find char within 127 of start
                ;           2 ¡ found wrong char
                ;.Y holds screen size
                ;
petent          ¡   *       ;entry for PET
                lda #$80    ;hi order of screen addr ($8000)
                bne vicent‡3 ;skips to load .Y with 0
                ;
vicent          ¡   *       ;entry for VIC
                lda $288    ;loads high byte of screen addr.
                ;($288 ¡ screen page on the VIC)
                ldy #0      ;and now load the lo byte
                ;
detent          ¡   *       ;on entry, screen addr in .A & .Y
                tax         ;save .A reg
                jsr $ffcc   ;restore default I/O devices (ie. screen)
                lda #19     ;reset screen controller window
                jsr $ffd2   ;by printing cursor home
                jsr $ffd2   ;twice
                lda #147    ;clear screen
                jsr $ffd2
                lda #17     ;do a cursor down
                jsr $ffd2
                lda #" 0"   ;the char we're going to look for
                jsr $ffd2
                ;
                lda 1       ;save (0) – we're going to
                pha         ;use it as a pointer
                lda 2
                pha
```

```
                sty   1       ;now set the pointer
                stx   2
                ldy   #1      ;now we're set
                ;
detloop         lda   (1),y   ;get the character
                cmp   #32     ;is it a space?
                bne   dettest ;no - test it
                iny           ;yes - prepare for next char spc
                bpl   detloop ;have we done this too many times?
                pla           ;yes - restore $0001 and $0002
                sta   2
                pla
                sta   1
                ldx   #1      ;return error code #1
                rts           ;and finish
                ;
dettest         cmp   #48     ;is the charactr just found" 0" ?
                beq   detsize ;yes - we've got the screen size
                pla           ;no - restore 1 & 2
                sta   2
                pla
                sta   1
                ldx   #2      ;and return with error #2
                rts
                ;
detsize         pla           ;we've got the size in .Y
                sta   2       ;restore 1 & 2
                pla
                sta   1
                lda   #147    ;tidy up by clearing screen
                jsr   $ffd2
                ldx   #0      ;return the all OK
                rts           ;screen len in .Y
```

Listing Two: Hex Dump.

```
.: 1000  a9  80  c0  03  ad  88  02  a0
.: 1008  00  aa  20  cc  ff  a9  13  20
.: 1010  d2  ff  20  d2  ff  a9  93  20
.: 1018  d2  ff  a9  11  20  d2  ff  a9
.: 1020  30  20  d2  ff  a5  01  48  a5
.: 1028  02  48  84  01  86  02  a0  01
.: 1030  b1  01  c9  20  d0  0c  c8  10
.: 1038  f7  68  85  02  68  85  01  a2
.: 1040  01  60  c9  30  f0  09  68  85
.: 1048  02  68  85  01  a2  02  60  68
.: 1050  85  02  68  85  01  a9  93  20
.: 1058  d2  ff  a2  00  60  xx  xx  xx
```

Editor's Note

The Commodore 64 uses locations 0 and 1 to do bank switching. This routine on the 64 might cause some adverse side effects. To avoid potential muck, change the routine's use of locations 1 & 2 to, say, 98 ($62) & 99 ($63) (Floating Point Accumulator). You'll also need another entry point (ie. c64ent) to adjust the screen start address. . . see your memory map.

Computer Technology

Robert Lovelace
Georgetown, Ont.

Robert Lovelace is Coordinator of Computer Applications and Development at Sheridan College of Applied Arts and Technology in Oakville, Ontario. The following is the opener for another regular Transactor column that will document the experiments and results of hardware and software activities at the post-diploma level at Sheridan.

Training in the microcomputer field takes several forms at Sheridan College. The two distinct orientations are business data processing and electronic computer technology. Microcomputer Management and Computer Science Technology provide training for all aspects of the business computing field. The courses in Electronics Technology and Microcomputer Technician emphasize machine level activity including design and interfacing skills. This traditional approach to computing education, the separation of hardware and software, is being enhanced with the addition of a new program in the School of Applied Science and Technology. "Microcomputer Applications" is a post diploma program for graduates of computer science or computer science technology courses at universities and colleges.

Programmers upon demonstrating their proficiency in high level and machine or assembly level programming will be instructed in all aspects of electronics, microcomputer architecture, interfacing and systems design. The goal of this program is to provide industry with a functional member of the design team who will fill the gap between the hardware and software experts. Industry advisors from Mitel, Litton, BMB Compuscience, Bell Northern, Ed. Agnew Assoc. and others have contributed significantly to the goals and curriculum of this program.

My intent is to wander through the courseware of this new program to expose to you some of its more interesting aspects. We will take a peek in several different labs on campus to investigate the uses of microcomputers in technical education and engineering. Some of the topics we will examine include CAI programs in electronics and assembler programming. Digital electronics, Telidon, building your own microcomputer trainer, Sheridan's custom CBM interface breadboarding system, our development lab and computer graphics will be discussed as well. Hopefully there will be a balance of hardware, software and philosophy in this coverage. This column could be the start of an information exchange for hobbyists and teachers of computer technology with your help.

It appears that Sheridan College has made a commitment to Commodore microcomputers and people often ask why. The answer; Why not? There are two sound reasons for this; first the hardware and second the software.

Entry level programming skills are best developed using a hands–on type of instruction. This means that to expose the largest number of students to computers, an institution needs a great number of machines unless time is scheduled through the evenings. Commodores three for two deal to educational institutions makes this possible and most low end personal computers perform equally well in this instructional mode. It is essential to expose students to other systems but this should only occur after some proficiency has been developed on one type. In the early stages a reduction in the number of variables will improve the learning experience.

The architecture of the machine allows for many interesting avenues of experiential learning. The built in user port, the IEEE–488 interface, the cassette ports and the memory expansion port give the user easy access to the real world. Tape recorders, slide projectors, analytical instruments can all be connected to the PET with minimal hardware and programmed in BASIC. Now math teachers can study number systems dynamically and interactively using lights on the lines of the user port. Physics teachers can time physical events with the built in timer, and electronics testing can be automated with IEEE–488 compatible instruments. Jim Butterfields IEEE Watch is an example of a simple testing program that allows you to observe the activity on the IEEE bus.

One of the best software resources available is the Toronto PET Users Group and their library of public domain software. At their meetings you will find some of the most knowledgable people in the computer field. Top this with some excellent educational software and the decision to stay with Commodore equipment becomes obvious.

In the next issue we'll look at Sheridan's development lab. To test the response of a control system, the *real* devices are first simulated with hardware. This allows one to predict potential problems thus reducing valuable construction time and expense.

Power On/Error Indicator For 4040 Disks

In 1980, Andrew Chiu of BMB Compuscience designed an add on circuit that offered 4040 users one of the niceties of owning an 8050. No, not more storage space, but rather an LED circuit that indicates power–on and doubles as an error indicator. Since this accessory is no longer being sold, BMB and Andrew have kindly given The Transactor permission to publish it here.

How many times have you sat down if front of your computer and reached around the back of your 4040 only to find that you're switching it off rather than on? While you feel how hot the cover is, you're thinking back and counting the hours your disk has been left on. Sound familiar? You're probably also thinking how many times you've walked by your disk and could have turned it off had you noticed. With this circuit, that uneasy feeling you're shaking off need never happen again.

The circuit uses a tri–colour LED. Like the 8050, it glows green continuously (indicating power on), and turns red when an error occurs (the third colour is "off"). The buzzer is also activated by an error, but this part of the circuit is optional.

It's incredibly easy to build. In fact, it'll take you longer to read this article and drive to Radio Shack for the parts than it will to put this thing together. All you need is one tri–colour LED, an SN 7400 (or any chip with 5 inverters on it) and the optional "piezo" buzzer. If you won't be needing the LED that came with your 4040, you can use the connector from it. To remove the factory assembly, you'll need to pry off the locking ring that holds in the LED. Once off, simply push the LED out from the front. Leave the mounting clip in place.

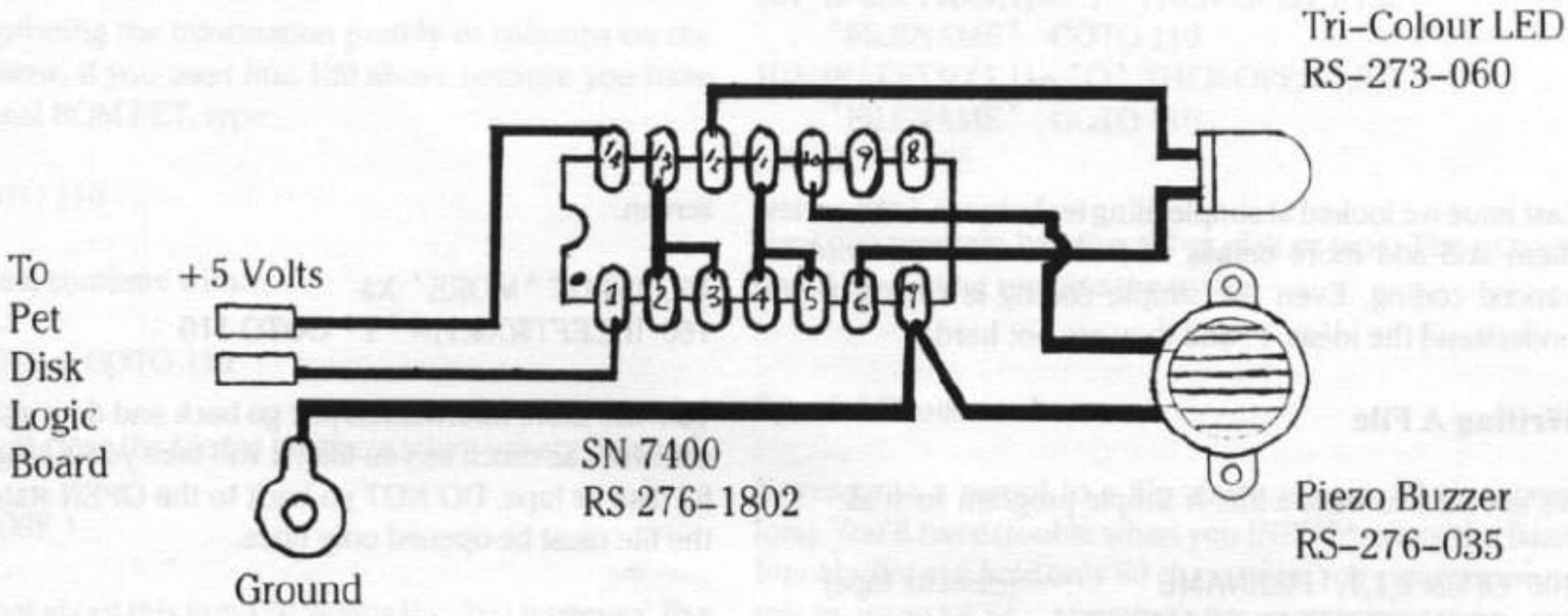
For a professional touch, use an eyelet connector for ground. You don't really need a circuit board. . . just use some electrical tape to cover up the pins and let the whole assembly sit on the bottom of the housing. If you choose to include the buzzer, you can tape it to the back of the chip.

The longer lead of the LED must be connected to pin 6 of the SN 7400 or the indicator will continuously glow red and turn green for errors. We suggest that the LED be connected to the IC with a pair of wires about 20 inches long. This way the leads to the logic board connector and the ground lead can be kept short. Also, with the buzzer towards the back, it's a little less arrogant on those quiet nights.

Once completed, you're ready for the acid test. Connect the ground lead to any logic board mounting screw (there's one just to the right of the pin connector). Connect the input lead next. The +5 volt pin is the one closest to the right of the unit when looking from the front. Turn your disk drive on. Any smoke? No? Good. If the LED is glowing green, you've passed test 1. Now try sending an error to the error channel:

```
open 1, 8, 15 : print#1, " jiberrish"
```

This will cause a ?SYNTAX ERROR and the LED should glow red. Read the error channel and it should go back to green. If all this works, slide the locking ring over the LED, insert it back into the mounting clip, and push the locking ring into place. You're done!



More About Filing

Jim Butterfield
Toronto, Ont.

Last issue we looked at simple filing techniques. Let's review them and add more details so that we can use more advanced coding. Even the simple coding is clearer if you understand the ideas. . . and they are not hard.

Writing A File

It's not hard to write a file. A simple program such as:

```
100 OPEN 2,1,1,"FILENAME"          (cassette tape)
100 OPEN 2,8,3,"0:FILENAME,S,W"    (disk)
100 DOPEN#2,"FILENAME",W           (disk, Basic 4.0)
. . .then. . .
110 INPUT "NAME";N$
120 PRINT#2,N$;CHR$(13);
```

The CHR$(13) is there to eliminate an unwanted "Linefeed" character. You don't need it for cassette tape or Basic 4.0; but it never hurts. When you use it, don't forget the semicolons.

```
130 INPUT "AGE";A
140 PRINT#2,A;CHR$(13);
```

Although we're writing a number, keep in mind that it will be written as characters, just as it would appear on the screen.

```
150 INPUT "MORE";X$
160 IF LEFT$(X$,1)="Y" GOTO 110
```

To write more information just go back and do it again. You can write as much as you like; it will take you a long time to fill disk or tape. DO NOT go back to the OPEN statement – the file must be opened only once.

```
170 CLOSE 2
```

You must say CLOSE, not only for neatness. As you write the information, it does not go to disk or tape immediately. It's collected in a buffer until there are enough characters to write. For tape, that's 191 data characters; for disk, it's 254 data characters. If you forget to say CLOSE, these characters will never be written, and the file won't be wrapped up correctly.

We don't know how many records we have written. It might be just one name and age, it might be five hundred. That doesn't matter; we'll be able to find out when we read the information back.

Reading The File

```
100 OPEN 1,1,0,"FILENAME"    (tape)
100 OPEN 1,8,2,"FILENAME"    (disk)
100 DOPEN#1,"FILENAME"       (disk with Basic 4.0)
...then...
110 INPUT#1,N$               (get name)
```

The following line is for Original Basic ROMs only. If your PET powers up with the message:

*** COMMODORE BASIC ***

including asterisks, you have an early model machine with slightly different logic. You need to code:

```
120 IF ST<>0 GOTO 200(Original ROM only)
```

More about that in a moment. Continue:

```
130 INPUT#1, A
140 PRINT"NAME:";LEFT$(N$+"",6);
150 PRINT"AGE:";RIGHT$("" +STR$(A),3)
```

We are printing the information prettily in columns on the screen. Now, if you used line 120 above because you have an Original ROM PET, type:

```
160 GOTO 110
```

Otherwise, continue with:

```
160 IF ST=0 GOTO 110
```

We should close the file for neatness when we are finished:

```
200 CLOSE 1
```

Now: what about this line 120 versus line 160 business? The very first PETs had a different logic from later models. Here's how the first PETs did it:

> "Try for the data. If it's not there because we're at the end, set ST equal to 64."

This has been changed in all later PETs, VICs and Commodore 64s to work this way:

> "If this is the last item of data, set ST equal to 64."

So the value in ST got signalled a little earlier – *with* the last item instead of *after* the last item.

The newer ROMs are much more standardized. On the early machines, tape worked a little unevenly and disk couldn't be connected. For this reason among many others, I like to urge owners of Original ROM PETs to get the new chip set that will make their machines more up to date.

Coding Comments

The first value in the OPEN statement is a number of your choice. It's a convenience number for you – the tape or disk has no idea if you are using 1, 2, or 50 and it obviously doesn't matter. After the OPEN, it's the only number you will use to refer to the file. It's called the Logical File Number.

There's a super bonus in the Logical File Number system, and it's visible in the coding above. Once we have opened the file, we don't care what the device is: our program will be the same in either case. In fact, we could change the input program above to select tape or disk:

```
100 INPUT"TAPE OR DISK";X$
101 IF LEFT$(X$,1)="T" THEN OPEN 1,1,0,
    "FILENAME":GOTO 110
102 IF LEFT$(X$,1)="D" THEN OPEN 1,8,2,
    "FILENAME":GOTO 110
103 GOTO 100
```

Now one program handles either disk or tape. The remainder of the coding remains the same.

Special Problem Areas

Never write a record to a file which is over 80 characters long. You'll have trouble when you INPUT#, since the Basic Input buffer can hold only 80 characters. You can get around this by using GET#... but that's time consuming.

Watch for special characters that can wreck your Input. The rules are the same as for file INPUT# as for keyboard INPUT... the dangerous characters are the comma, the colon, and the quotation marks. Once again, you can work around this problem with a GET# statement. But it's much easier to avoid the special characters and reap the benefits of the powerful INPUT# command.

Conclusion

The fine points are not that difficult. Think of the rules, and filing data will give you no trouble.

Translator Utility

Bill MacLean
Milton, Ont.

Many computer systems function very well when operating under their own well defined environment. However, once the computer steps outside it's own confines and communicates with foreign devices it may discover that it does not speak the same tongue as the other device. For example, PET ASCII and true ASCII code differ and are often converted from one to the other by a table conversion (see the last Transactor).

One common use of this translator is to drive several different printers with the same software. For example, to drive an EPSON printer with software designed for the Commodore matrix printers we would convert not only the PET–ASCII to true ASCII but would also convert the printer control codes. For enhanced print we would map the PET character –1– to character –14– for the EPSON. For paging we would map the PET [clr screen] character –147– to –0– (ignore it) and the PET [home] character –19– to character –12– (top of form) for the EPSON.

Similar translations are useful in a communications environment. When speaking to other computers, some of the PET cursor control characters are completely misunderstood by the other system. . . sometimes with tragic ends.

Other applications include decoding true ASCII disk files generated by one program for use with another that deals with PET ASCII only. The sample program demonstrates an encryption technique allowing one to encode information within the software. At some later time, the same utility would then decipher the strings by simply reversing the process.

The routine was originally written to help generate multi-–lingual software where additional characters are added to the alphabet and must be sorted into the original 26 characters (did you know that the french have 4 'e's between 'd' and 'f'). We used machine code for speed, and the code is completely relocatable and will accommodate a table located anywhere in memory. As shown, it starts at $7d00 which allows room for two 256 byte tables to be placed at $7e00 and $7f00. The sample program protects the routine from BASIC with POKE 53, 125 : CLR.

To use the routine, you simply call the code with a "sys" followed by two arguments:

sys ad, a$, ta%

"ad" is the start address of the machine code

"ta%" is the start address of the table, note that ta% is a decimal number and has a value between –32767 and +32767. If the table is located anywhere in the bottom 32k of the PET the absolute address can be used. However, if the table is located above $8000 (32768) then then two's compliment of the address must be used. If the table is at $9000 (36864) then ad% = 36864 – 65535.

"a$" is the name of the string to be translated, and must reside in the computer's memory someplace.

The result of the translation is placed into the called string variable.

Source Code

```
                BASIC:      4.0         2.0

                0 chkcom  = $bef5       ;$cdf8
                1 fndvar  = $bd98       ;$cc9f
                2 discrd  = $c7b5       ;$d57d
                3 fltpt   = $c92d       ;$d6d2
                4 toerr   = $bf00       ;$ce03

               10         * = $7000
              100 translat jsr   chkcom
              110          jsr   fndvar
              120          ldx   $07
              130          bpl   nulvar
              140          jsr   discrd
              150          sta   $00
              160          stx   $01
              170          sty   $02
              180          jsr   chkcom
              190          jsr   fndvar
              200          jsr   fltpt
              210          ldy   #$00
              220 tr1      lda   ($01),y
              230          sty   $0f
              240          tay
              250          lda   ($11),y
              260          ldy   $0f
              270          sta   ($01),y
              280          iny
              290          cpy   $00
              300          bne   tr1
              310          rts
              320 nulvar   jmp   toerr
              330 .end
```

A Nifty Encryption Routine

```
100 poke 53, 125 : clr : gosub 1000     : rem protect space & set up m.c.
110 b=7*4096 + 13*256                   : rem address of m.c. ($7d00)
120 a=7*4096 + 14*256                   : rem address of table 1 ($7e00)
130 for i=0 to 255 : poke a+i, i : next : rem make table
140 ke=17                               : rem encryption key
150 input " some string (eg, name)" ; a$
160 sys b, a$, a+ke : print a$          : rem encode a$
170 sys b, a$, a-ke : print a$          : rem a$ ok again
180 print : goto 150
```

BASIC Loader (BASIC 4.0 version)

```
1000 for j=32000 to 32049 : read x : poke j, x : next : return
1010 data  32, 245, 190,  32, 152, 189, 166,   7,  16,  37
1020 data  32, 181, 199, 133,   0, 134,   1, 132,   2,  32
1030 data 245, 190,  32, 152, 189,  32,  45, 201, 160,   0
1040 data 177,   1, 132,  15, 168, 177,  17, 164,  15, 145
1050 data   1, 200, 196,   0, 208, 240,  96,  76,   0, 191
```

Making Friends With Sid

Paul Higginbottom
Toronto, Ont.

The synthesizer chip in your Commodore–64 computer is affectionately known as Sid. Sid is in fact an acronym for **S**ound **I**interface **D**evice. I doubt that many people realise just how powerful this chip is, but I intend to unleash some of its power for you. If you read some of the documentation for the Commodore–64 about its sound capabilities and are new to synthesizer jargon (as I was), you probably thought to yourself, "I'm never going to figure that out!" Well, I am the sort of person who gets more determined to figure something out, when it seems harder than ever to do so. So, step by step, I, like any beginner, set about learning how to control the Sid's sound capability.

The Jargon

If your mind is like mine and tends to go blank when confronted with a barrage of alien jargon about something, then hopefully I can gently "break you in" with the terms associated with music synthesis using Sid.

The Sid chip is comprised of three sections essentially:

1) Oscillator section

2) Envelope section

3) Filter section

There are a few other bits and pieces, but more on those later.

Sid has three voices. That means to you and me, that up to three tones can be played at the same time.

Each voice is seperately controlled by its frequency (the pitch of the tone), and more importantly, its envelope.

The envelope of a voice, determines how its volume rises, sustains, and falls, like a musical instrument, or other sounds we hear in our lives. For example, a violinist, will maybe play a note by pulling the bow across a string slowly at first (the volume starting out low), and as the player starts to increase the speed and pressure of the bow on the string, so the volume increases, and as the player ends the note, he or she slows the rate and pressure of the bow again, and the volume fades away to silence. With a single violinist, the tone may fade away rather abruptly, but I'm sure you've heard this rising and falling effect of volume, with a piece of orchestration (many string instuments). That is one example of an envelope.

If we consider another example to allow you to grasp different types, think of hitting a cymbal. The rise to its maximum volume is almost instant, as the CRASH of the cymbal begins, and from that point, the sound simply fades

away slowly to silence again. An example of a cymbal type of sound that does rise slowly first and then fade away would be a wave approaching the beach. You hear the slowly increasing volume of the wave moving up the beach, then as the wave trips over itself and hits the beach the loudest part of the noise is heard, and then the sound fades away as the wave slides up the beach, and the next one approaches again.

Well, enough of the examples, back to the technical stuff. This "behaviour" of the volume (or amplitude) of a voice, can be defined in 4 parts, and this terminology is common amongst professional synthesizers costing many times the prices of your Commodore–64 computer!

The Four Parts Of An Envelope

You may have noticed by now, that to define this changing in volume, we simply need to define the TIME it takes for a sound to go from one volume, to get to another volume. For example, the voilin might have taken half a second to go from no volume (silence) to its maximum volume, and then 2 seconds to fade away again (silence again). The cymbal took no time to reach its maximum volume (starts with the CRASH), but 10 seconds to fade away. The wave is different again, in that it might take 5 or so seconds to build up to maximum volume (as it moves up the beach), and then only 1 second to die away (as the wave falls over and crashes on the beach).

Part 1 – ATTACK This is the time taken to go from silence (0 volume) to the maximum volume Sid is set to.

Part 2 – DECAY This is the time taken to go from the maximum volume Sid is set to, to a given "mid–point" volume, or, sustained level of volume.

Part 3 – SUSTAIN This is not a time value, but is a level of volume the voice sustains at after the ATTACK and DECAY.

Part 4 – RELEASE This is the time taken to go from the sustained volume to silence once again.

In those definitions, I mentioned "the maximum volume Sid is set to", and that is the maximum overall volume (just like the overall volume control on your television or stereo).

How We Control Sid

Before I go any further, I want to explain how we actually tell Sid exactly what weird and wonderful sounds we want it to make (so we can drive everyone crazy!)

The Sid chip has an amount of memory in it, and simply by putting numbers into those memories, we give Sid all the information it needs to produce an infinite number of sounds. We put numbers into memories with the BASIC command "POKE". We give the POKE command two numbers; the memory number (or "address"), and the number we want to put into that memory (one memory location can hold any whole number between 0 and 255).

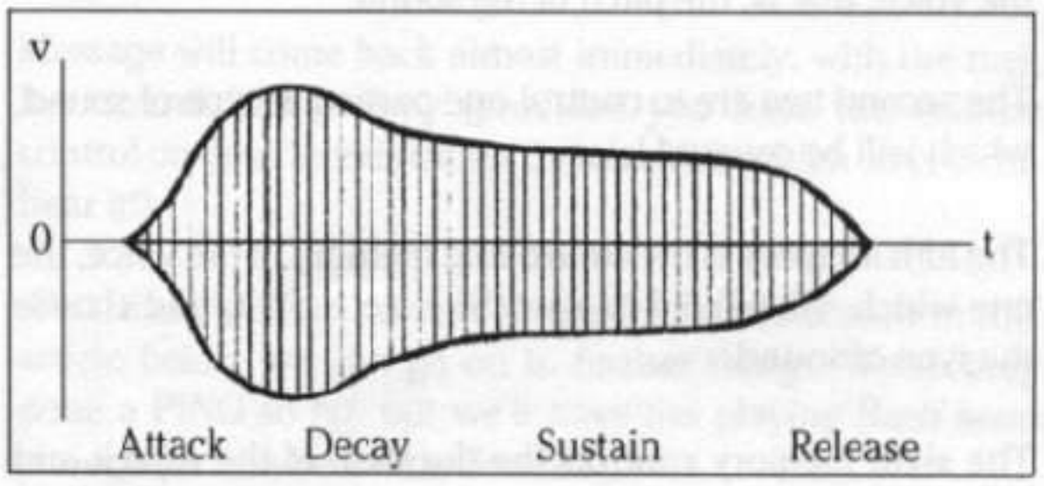
Sids address is quite a big number. He starts at 54272, and he occupies that memory location and the next 28 also, up to 54300.

I want to show that it really is not that difficult to train Sid, and that you don't have to be a genius at programming.

Making Your First Beep

To make a noise, we must do 4 things:

1) Set the maximum overall volume
2) Set the envelope of the voice we wish to use
3) Set the frequency of the voice to the desired pitch
4) And only then "tell" Sid to do it.



I put quotes around "tell" in part 4, because I want to examine that closer. When we tell Sid to make a sound, we tell it to firstly do the ATTACK (rise up to maximum volume) and then the DECAY (go down) to a SUSTAINed level of volume. When we tell Sid to do that part, the noise will stay at the SUSTAINed level of volume forever if you don't tell it to go on, and do the last part; the RELEASE (go down from the sustained level of volume, to nothing).

So to recap, we tell Sid to do the ATTACK–DECAY–SUSTAIN part first, and then when we're ready, we tell it to finish the envelope with the RELEASE part.

You could get a person to demonstrate this for you. Ask them to take a DEEP breath when you tap them on the shoulder and then hum a note at first quietly buildin up to a loud level, and the going down to a comfortable level. You have mad3e a person do the Attack–Decay–Sustain part of an envelope. I said the sound will continue indefinitely if you don't tell it to release, so when you tap the person again on the shoulder, they can slowly quieten their hum down to nothing. Of course if you decide to make them sustain for two long, they'll go blue in the face, and pass out! (Also, you may want them to stop before they get to the release, because their hum is so obnoxious! (Fortunately, you can also do this with the Commodore–64!)

For now, let's just concentrate on ONE voice. Each voice has 7 memories inside Sid, to control it. Voice 1's memories are in fact, the first 7 memories, voice 2, the second 7, and voice 3, the next 7. That, if you've been doing your math, is the first 21 memories in Sid. The other 8 (there are 29 in all) are for the filter section which I haven't talked about yet, and other bits and pieces, including the overall volume control which I have mentioned).

The 7 memories for each voice are all organised the same way, for example, the first two of each block of 7, control the frequency (pitch) of the voice.

The 7 Memories For A Voice

The first two as I just mentioned, control the frequency of the voice, that is, the pitch of the sound.

The second two are to control one particular type of sound, which will be covered later

The fifth memory is the controlling memory of the voice, the one which will tell Sid to start the note, stop it, and choose the type of sound.

The sixth memory controls the duration of the Attack and Decay.

The seventh memory controls the Sustain level, and the Release time.

The fifth of the seven I just described, I will now explain further. I mentioned there that apart from telling Sid to play the envelope, it also controls the type of sound. (Another piece of jargon coming up!)The type of sound is known as the Waveform. You are probably aware that sound is comprised of air being compressed and stretched. By, for example, a speaker cone, which moves in and out, and the speed (the FREQUENCY) at which it moves in and out, determines the pitch.

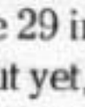
The way in which air is compressed and stretched is cyclic (repeats itself), and this cycle is known as the waveform.

Sid allows you to choose from one of four waveforms, and it is the fifth of the seven memories in each voice, which you set to tell Sid which waveform you wish to use.

The Waveforms

Triangular (shaped like this:)This waveform, due its smoothness produces a mellow, soft flute–like sound (very pleasant to the ear!)

Sawtooth (shaped like this:)This waveform, due to its abrupt ending produces a brighter, brass–like sound.

Variable width pulse (shaped anything from this:  , to this: , or this: [which if you look is simply the first one upside–down])

This waveform as you can see from the description in parenthesese can be varied, but is essentially an ON and OFF waveform, and as such is very abrupt and produces anything from a hollow, organ–like sound, to a very quiet, reedy sound. As you can see from the symbols of this waveform, it is comprised of pulse, or varying widths (hence the name), and memories 2 and 3 which I mentioned earlier were for a specific type of sound, do in fact control the width of the pulse when this waveform is chosen. Of course, memories 2 and 3 have no effect when any other waveform is selected.

Noise – I won't try to do a little drawing of this waveform, since it is in fact a RANDOM waveform, and has no defined harmonic qualities, but because the frequency can be altered, will produce any sound from a hiss (like you hear from poor quality cassette recorders), to a low rumble (good for special effects in games).

Please note the format of the memory locations used. I mentioned that the first TWO bytes are used to select the frequency, but did not say how one would know what values to put in one and the other. The easiest way to look at it, I would expect would be thus:

Since one memory location can only contain a number from 0 to 255, to represent larger numbers, they are stored as 0 to

255 in the first byte (known as the low byte), and multiples of 256 added to this in the second byte (known as the high byte), which means two bytes can hold a number from 0 (0 in both locations), to 65535 (255 in the first one, plus, 255 times 256 in the second one).

The ATTACK–DECAY memory, and the SUSTAIN–RELEASE memory are comprised as follows:

The ATTACK, DECAY, SUSTAIN, RELEASE parameters can all be one of 0 to 15. To form the ATTACK–DECAY value for memory location 6 in a voices 7 memory locations, simply multiply the ATTACK by 16 and add the DECAY value. This again gives a combined value from 0 (0*16+0) to 255 (15*16+15).

The control register works differently still. The value is calculated as follows:

Add 1 to begin Attack–Decay–Sustain cycle; don't add 1 to begin the Release cycle.

Add 16 to select triangular wave form, 32 for sawtooth, 64 for variable width pulse, or 128 for noise waveform.

There are other parts to add to this value, but they won't be covered here.

The Beep Program

Turn on your Commodore–64, and type in the following program:

```
10  sid=54272
20  for i=0 to 28 : poke sid+i, 0 : next
30  poke sid+24, 15
40  poke sid+  1, 20
50  poke sid+  5, 0*16+0
60  poke sid+  6, 15*16+9
70  poke sid+  4, 1+16
80  poke sid+  4, 16
```

Description Of The Program

Line 10 defines a variable SID, to the start of Sid's memory locations.

Line 20 should be included in all of your sound programs, and is a FOR..NEXT loop to simply set all of Sid's memory locations to 0 to ensure that no previous programs will affect our efforts.

Line 30 sets memory location 24 in Sid, to 15. Register 24 controls the overall volume of Sid (and some other things which need not be known here), and 15 is the maximum volume (from 0 to 15).

Line 40 sets the upper byte of the frequency value of voice 1 to 20, which means a setting of 0+20*256 = 5120.

Line 50 sets the ATTACK value of voice 1 to 0, and the DECAY value also to 0, which means when we tell Sid to do its ATTACK–DECAY–SUSTAIN cycle, it will simply go straight to the SUSTAIN volume, since we've told it not to do any ATTACK or DECAY at all.

Line 60 sets the SUSTAIN value of voice 1 to 15 (maximum volume), and the RELEASE value also to 9, which means when we tell Sid to do its RELEASE cycle, it will take about three quarters of a second to fade away to nothing.

Line 70 sets the control register of voice 1 to do the Attack–Decay–Sustain sequence, with the triangular waveform selected (+16).

Line 80 sets the control register of voice 1 to do the Release part of the envelope, again with the triangular waveform selected (+16).

Having typed in this program, type:

```
RUN
```

And the familiar:

```
READY.
□
```

Message will come back almost immediately, with the mellow sound fading away (provided you have the volume control on your television set up reasonably high so you can hear it!)

Well, there's a LOT of new things for you to absorb in this article before we can go on to further things. We've only done a PING so far, but we'll have Sid playing Bach soon enough (or maybe a little Genesis?)

Commodore 64 Piano/Organ Program

Want to hear more from SID? Then type in this short program also by Paul Higginbottom!

The first half of the program sets up the screen, the keyboard, and the chip itself. Lines 90–140 simply draw a piano

keyboard on the screen, so if you don't get it right from the listing, don't worry 'bout it. . . they don't affect the operation of the program. Lines 150–180 are instructions.

Line 190 sets "s" equal to the base address of the SID chip. 192 and 193 declare some other default values. After that comes the keyboard. You'll notice that Paul uses translation array techniques discussed in another article in last issue. 240 erases the message printed by 180, and 260–280 resets the SID chip. Line 290 is "REMed" out for some reason. Try removing the REM to see what it does.

Once all set up, lines 300 on do all the work! Lines 500–570 are looking for the function keys at the right. 580 detects the "space bar" . To clear everything and start again, hit SHIFT & CLR (line 590).

That's about it. The program is pretty easy to follow around (like most Higginbottom programs), so if you can assimilate this one, SID theory will become second nature.

If you want, add the lines following the first listing. The "X" command will execute the song stored in the data statements.

Yet another Higginbottom program, SIDMON 1.0, appears in Volume 1, Issue 3, of Commodore's Power Play magazine. SIDMON 1.0 is, as the name suggests, a SID Monitor that lets you change the envelope values, waveforms, volume, frequency, filter and special effects sections of the SID chip. The program is good for experimenting when you want to find a specific sound for a desired effect. Paul's Organ 64 program allows for more dynamic playing although not all of SID's features are implemented (hint hint).

```
90 print"  
100 print"  
110 print"  
120 print"  
130 print"  
140 print"  q | w | e | r | t | y | u | i | o | p | @ | * | ↑ "
```

```
150 print " [q][r] space [R] to select '1' or '3' voice mode
160 print " [q][r] f1 [ ] f3 [R] octave up/down
170 print " [q][r] f5 [R] next 'musical' waveform "
175 print " [q][r] f7 [R] select noise waveform "
176 print " [q][r] a [ ] s [R] tuning controls down/up "
177 print " [q][r] m [R] select mode: piano, organ, hold, glide
180 print " [q]*** setting up frequency table *** "
190 s=13*4096+1024
192 at=0 : de=12 : su=0 : re=0 : sr=su*16+re : ad=at*16+de : wv=17 : w=0 : oc=1 : tu=1
193 c7=7 : c4=4 : c0=0 : c1=1 : c3=3 : ky=197 : nk=64 : c2=2 : c5=5 : c6=6 : hb=256 : co=2 ^ (1/12)
195 dim fr(22,4), k(255)
200 for i=0 to 28 : poke s+i, c0 : next : for i=0 to 4 : a(i)=2 ^ i : next
205 f1=7040 : for i=22 to 1 step-1 : f=f1*5.8+30 : for j=0 to 4 : fr(i,j)=f/a(j) : next
215 f1=f1/co : next
220 k$=" q2w3er5t6y7ui9o0p@-*\↑"
230 for i=1 to len(k$) : k(asc(mid$(k$,i)))=i : next
240 print " [Q]                                                    "
260 for i=0 to 2 : t=i*7 : poke s+c5+t, ad : poke s+c6+t, sr : poke s+c3+t, 15 : next
280 poke s+24, 15
290 rem poke s+14, 0 : poke s+15, 0 : poke s+19, 0 : poke s+20, 240 : poke s+18, 17 : poke s+24, 255
300 get a$ : if a$="" then 300
310 se=k(asc(a$)) : fr=fr(se,oc) : if fr=c0 then 500
320 t=v*c7 : cr=s+t+c4 : fr=fr*tu
330 og=0 : ig=30 : poke cr, z
335 if md=c3 then 1000
340 poke s+t, fr-int(fr/hb)*hb
350 poke s+t+c1, fr/hb
370 poke cr, wv
380 if p=c1 then v=v+c1 : if v=c3 then v=c0
390 if (md<>c1) then 300
395 if peek(ky)<>nk then 395
400 pokecr, 8 : goto 300
500 if a$=" [E] " then oc=oc-1-(oc=0)
510 if a$=" [F] " then oc=oc+1+(oc=4)
540 if a$=" [G] " then wv=((((wv and 254)*2) and 127) or 1)-16*(wv>33)
570 if a$=" [H] "then wv=129
580 if a$="  " then p=1-p
581 if a$="s" then tu=tu+.005
582 if a$="a" and tu>0 then tu=tu-.005
587 if a$="m" then md=md+1 : goto 900
590 if a$=" [S] " then 200
600 goto 300
900 if md=0 then for i=0 to 2 : t=i*7 : poke s+5+t, ad : pokes+6+t, z : next : goto 300
910 if (md=1) or (md=2) then for i=0 to 2 : t=i*7 : poke s+5+t, z : poke s+6+t, 240 : next : goto 300
920 if md>c3 then md=0 : goto 900
930 goto 300
1000 poke cr, wv
1005 for i=lf to fr step(fr-lf)/8
1010 poke s+t, i-int(i/hb)*hb
1020 poke s+t+c1, i/hb
1030 next
1035 poke s+t, fr-int(fr/hb)*hb
1036 poke s+t+c1, fr/hb
1040 ls=se : lf=fr : goto 380
```

```
585 if a$ = "x" then 10000

9000 rem tune 1: (tunes end with '0')
9002 data 20, 17, 15, 17, 13, 17, 12, 17, 10, 17,  8, 17
9005 data  6, 17, 15, 13, 15, 15, 13, 15, 12, 15, 10, 15
9006 data  8, 15,  6, 15,  5, 15, 13, 12, 13, 13, 12, 13
9008 data 10, 13,  8, 13,  6, 13,  5, 13,  4, 13, 12, 10
9010 data 12, 12, 10, 12,  9, 12,  7, 12,  5, 12,  3, 12
9020 data  1, 12, 10,  8, 10,  0
9600 rem tune 2:
9605 data 20, 12,  8, 10, 12, 15, 13, 13, 17, 15, 15, 20
9610 data 19, 20, 15, 12,  8, 10, 12, 13, 15, 17, 15, 13
9620 data 12, 10, 12,  8,  7,  8, 10,  3,  7, 10, 13, 12
9630 data 10, 12,  8, 10, 12, 15, 13, 13, 17, 15, 15, 20
9640 data 19, 20, 15, 12,  8, 10, 12,  5, 15, 13, 12, 10
9650 data  8,  3,  8,  7,  8, 12, 15, 20, 15, 12,  8, 12
9660 data 15, 18, 15, 12,  8, 12, 15, 17, 13, 10,  7, 10
9670 data 13, 15, 12,  8,  5,  8, 12, 13, 10,  7,  3,  7
9680 data 10, 13, 12, 10,  8, 12, 15, 20,  0
9700 rem tune 3:
9705 data 20,  5, 10, 12, 13, 17, 13, 12, 13,  0
9800 rem add tunes here (-1, -1 = no more)
9999 data -1, -1

10000 read te : print : rem read 'tempo' (1st element of tune data)
10002                 : rem 20 in all of above
10005 read a : if a=z then print : goto 300
10006 print a;
10010 if a=-1 then restore : print : goto 300
10015 fr=f(a)/m : t=v*7 : cr=s+t+4
10020 poke s+5+t, z : poke s+6+t, z
10030 poke cr, 8 : poke cr, z
10040 poke s+t, fr-hb*int(fr/hb)
10050 poke s+1+t, fr/hb
10060 poke s+5+t, ad : poke s+6+t, sr
10070 poke cr, wv+1 : for i=1 to 50*at : next
10075 poke cr, wv
10080 if p=1 then v=v+1 : if v=3 then v=0
10090 for i=1 to te : next
10100 goto 10005
```

The 64 Skiffle Band

Jim Butterfield
Toronto, Ont.

Somewhere up in the hills of Commodore county, there's a jug band and skiffle group that gathers from time to time. I'd like to introduce you to the players, and show you how to program your Commodore 64 so as to hear them play.

This is a basic band playing basic music. . . so you can write your program in Basic. Later, you may want to change instruments and add to the group's repertoire. With the 64 sound chip, it's not hard to do.

We're going to be the bandleader, so we must set up the instruments and choose the type of play. . .Here we go:

```
100 print "MUSIC  (By Jim Butterfield)"
```

This identifies the maestro (that's me). Now, let's locate the

musicians:

```
110 l1=54272 : l2=54279 : l3=54286
120 h1=l1+1 : h2=l2+1 : h3=l3+1
130 v1=l1+4 : v2=l2+4 : v3=l3+4
```

Each musician stands in a "spot" marked by a number. As band director, we'll call up the music for each one by signalling each player's spot, using his "address". The L's and H's will be used to name the pitch of the notes; the V locations are used for the other stuff.

```
140 poke 54296, 15
```

This tells the listeners to turn up their hearing aids, so they can hear the group playing. The highest volume we can set is 15, and that's the one we pick.

```
150 poke v1+1, 9 : poke v1+2, 0
```

Player 1 is young Billy; he plays the hanging jugs. When he hits a jug with his corncob drumstick it will sound right away, and come out with a good clunking sound (that's the 9, for fast sound attack and fulsome decay). But he can't strum the jug or hold the sound, so the note will fade away soon after he plays (that's the 0, for no sound sustain or release).

```
160 poke v2+1, 36 : poke v2+2, 36
```

Player 2 is Bearcat Zeke, who plays the harmonisqueeze. When Zeke puffs into his instrument, it takes a moment for the sound to build up (we pick 36, for a medium speed attack and moderate decay). Once he winds up the note, he can hold it for quite a while (we set 36 for medium sustain and release).

```
170 poke v3+1, 18 : poke v3+2, 250
```

Old man Mose is in slot number 3; he plays a traditional gut bucket. When he strums the cord, he gets a fast, full sound (we'll pick 18 for fast attack and medium decay). Once the cord is in motion, it holds the sound for quite a while before it fades away (250 for a full slow sound decline).

```
180 t=ti
```

Let's check our digital watches so that we can get the band playing in time.

```
200 poke v1, 16 : poke v2, 32 : poke v3, 16
```

Here's where we give each instrument its individual sound. Billy and old man Mose have instruments that produced a clear, soft sound. For them, we pick a "triangle" waveform, code 16. Zeke's contraption makes a sharper, snarly sound; we give it an edge with the "sawtooth", code 32.

We're doing something else important here. By using *even* numbers, we're telling the boys: hands off the instruments. Billy, take your corncob away from the jugs; Zeke, don't blow into that thing; and old man Mose, hands off the string. Later, we'll tell them to "hit it" with an *odd* number to the same location.

It's time to play some music. Let's read the notes.

```
210 read s : if s=0 goto 290
```

First, let's get the timing. If the note timing is zero, we must be finished, because we can't play that fast. . .

```
220 read x1, y1, x2, y2, x3, y3
```

Let's get the notes. Each note is broken into two pieces; that's the way we feed it to the band. If the note is zero, we have nothing to play this time around.

```
230 if x1 then poke h1, x1 : poke l1, y1 : poke v1, 17
240 if x2 then poke h2, x2 : poke l2, y2 : poke v2, 33
250 if x3 then poke h3, x3 : poke l3, y3 : poke v3, 17
```

If a player has something to play (his note is not zero) we'll put his note into the proper slot, and then tell him to "hit it", by making his waveform number odd – remember line 200 where we told everyone hands off?

In fact, we're going to go back to 200 later and tell the fellows to let go. If we didn't, they wouldn't be able to play the next note. But first, let's do some timing. . .

```
260 t=t+s
```

Let's set the clock ahead so that we can tell when this note has been played long enough. This way, we can wait until the note has had its full time:

```
270 if t>ti goto 270
```

When the note has played, we'll go back and do it all again, with:

```
280 goto 200
```

And when the music is finished, we tell the band to take a break. . . until the next tune. Turn off the instruments. . .

```
290 for j=l1 to 54296 : poke j, 0 : next j
```

That's all very well, but we'll need some music to play. Here it comes:

```
300 data  10,  51,  97,   0,   0,   0,   0
310 data  10,  43,  52,   0,   0,   0,   0
320 data  20,  34,  75,  21, 154,   8, 147
330 data  20,  34,  75,  25, 177,   0,   0
340 data  10,  34,  75,  17,  37,   6, 108
350 data  10,  38, 126,   0,   0,   0,   0
360 data  10,  43,  52,  25, 177,   0,   0
370 data  10,  45, 198,   0,   0,   0,   0
400 data  20,  51,  97,  21, 154,   8, 147
410 data  20,  51,  97,  25, 177,   0,   0
420 data  20,  51,  97,  17,  37,   9, 159
430 data  20,  43,  52,  25, 177,  10, 205
440 data  20,  57, 172,  22, 227,  11, 114
450 data  20,  57, 172,  34,  75,   0,   0
460 data  20,  57, 172,  28, 214,   8, 147
470 data  10,   0,   0,  34,  75,   0,   0
480 data  10,  51,  97,   0,   0,   0,   0
500 data  20,  57, 172,  22, 227,  11, 114
510 data  10,   0,   0,  34,  75,   0,   0
520 data  10,  51,  97,   0,   0,   0,   0
530 data  10,  57, 172,  28, 214,  10, 205
540 data  10,  64, 188,   0,   0,   0,   0
550 data  10,  68, 149,  34,  75,   9, 159
560 data  10,  76, 252,   0,   0,   0,   0
600 data  20,  86, 105,  21, 154,   8, 147
610 data  20,   0,   0,  25, 177,   0,   0
620 data  20,   0,   0,  17,  37,   6, 108
630 data  10,  68, 149,  25, 177,   0,   0
640 data  10,  51,  97,   0,   0,   0,   0
650 data  20,  68, 149,  21, 154,   8, 147
660 data  20,   0,   0,  25, 177,   0,   0
670 data  20,   0,   0,  17,  37,   8,  23
680 data  10,  51,  97,  25, 177,   7,  53
690 data  10,  43,  52,   0,   0,   0,   0
700 data  20,  51,  97,  22, 227,   6, 108
710 data  20,   0,   0,  25, 177,   0,   0
720 data  20,   0,   0,  19,  63,   9, 159
730 data  10,  38, 126,  25, 177,   0,   0
740 data  10,  43, 100,   0,   0,   0,   0
750 data  20,  34,  75,  21, 154,   8, 147
760 data  20,   0,   0,  25, 177,   6, 108
770 data  20,   0,   0,  21, 154,   4,  73
780 data  20,   0,   0,   0,   0,   0,   0
790 data  0
```

That's the whole thing. You can add to the music, change the instruments, change the timing, or do whatever else you like.

If you'd like to listen to the instruments one or two at a time, you can silence any player by changing the appropriate line 230 – 250 to a REM line. You could delete the line, but REM is better – it saves typing when you want to bring the instrument back.

An easy way to change the speed of a tune is to change line 260: by multiplying or dividing variable S by an appropriate factor, the band can turn frantic or ease back into lazy playing.

If you change the waveform, remember that there are two places to do it: line 200 and the appropriate line of 230 – 250.

There are a couple of waveforms that we haven't used: we've stayed with triangle (16) and sawtooth (32) instruments. If you want to bring in fiddlin' Fran, she will need a pulse waveform (64), and you'll need to supply a *pulse width* by setting values into (for instrument 1) V1–1 and/or V1–2. Snaredrum Sammy will want a noise waveform (128). Don't forget to add 1 when you want the instrument played.

The group mostly plays southern music. But on a pleasant warm night when visitors stop by to listen, they might just try a northern tune. . .

```
300 data  20,  34,  75,  21, 154,   8, 147
310 data  20,  34,  75,  25, 177,   0,   0
320 data  20,  38, 126,  28, 214,   6, 108
330 data  20,  43,  52,  25, 177,   0,   0
340 data  20,  34,  75,  21, 154,   8, 147
350 data  20,  43, 180,  25, 177,   0,   0
360 data  20,  38, 126,  22, 227,   8,  23
370 data  20,   0,   0,  25, 177,   0,   0
400 data  20,  34,  75,  21, 154,   8, 147
410 data  20,  34,  75,  25, 177,   0,   0
420 data  20,  38, 126,  28, 214,   6, 108
430 data  20,  43,  52,  25, 177,   0,   0
440 data  20,  34,  75,  21, 154,   8, 147
450 data  20,   0,   0,  25, 177,   0,   0
460 data  20,  32,  94,  22, 227,   8,  23
470 data  20,   0,   0,  19,  63,   6, 108
500 data  20,  34,  75,  21, 154,   8, 147
510 data  20,  34,  75,  25, 177,   0,   0
520 data  20,  38, 126,  21, 154,   7, 163
530 data  20,  43,  52,  17,  37,   0,   0
540 data  20,  45, 198,  28, 214,   7,  53
550 data  20,  43,  52,  34,  75,   0,   0
560 data  20,  38, 116,  28, 214,   0, 206
570 data  20,  34,  75,  22, 227,   0,   0
600 data  20,  32,  94,  25, 177,   6, 108
610 data  20,  25, 177,  22, 227,   0,   0
620 data  20,  28, 214,  21, 154,   7,  53
630 data  20,  32,  94,  19,  63,   8,  23
640 data  20,  34,  75,  21, 154,   8, 147
650 data  20,   0,   0,  25, 177,   6, 108
660 data  20,  34,  75,  21, 154,   4,  73
670 data  20,   0,   0,   0,   0,   0,   0
700 data  0
```

Next time you're passing through the hills of 64, drop in, set a spell. . . and play along.

Commodore Assembler Editor For Basic–Aid

F. Arthur Cochrane
Beech Island, SC

The program presented here links with the Basic–Aid program as published in Transactor Volume 3 Issue 6 page 52. This program combined with Basic–Aid becomes a Commodore Assembler Editor.

After Basic–Aid was done it was noticed that Basic–Aid was very similar to the Commodore Assembler Editor that is supplied with the Assembler Development Package. The only thing Basic–Aid lacked was a method to GET and PUT the source code files to the disk. Since the source code for the Commodore Editor is supplied with the Development Package it is available for easy modification and use.

The program presented here changes the CHRGET Wedge routine to point to itself. A line that is input beginning with a line number is inserted into memory without being tokenized. Otherwise this program checks to see if the direct command is in its command list, and if it is the command is executed. If the command is not an editor command then a jump is made to Basic–Aid to let it check its commands.

Using Basic–Aid as part of an assembler editor allows the programmer all the features of Basic–Aid to be used in the editing of machine language source code. The most powerful of these is the ability to scroll through the code with the cursor control keys.

The program as presented here is for BASIC 4.0 and Basic–Aid at $7000, but it could be assembled to executed for BASIC 2.0 and Basic–Aid somewhere else.

Two notes on this program and Basic–Aid. The Commodore Assembler appears to use locations in the second cassette buffer that are also used by Basic–Aid and this program, so it is advised to use the KILL command to disable them before running the assembler (they can be enabled with a SYS after running the assembler). Also do not use the BASIC 4.0 CONCAT command with Basic–Aid enabled. This is because BASIC 4.0 builds DOS commands in the second cassette buffer before sending them to the disk. Basic–Aid uses locations in the second cassette buffer that will be overwritten by the building of the DOS command for the CONCAT statement, since it is the longest one built.

If the reader is lazy, the source code for Basic–Aid and this program are available from ATUG. Below is a list of the instructions, a hex dump and a BASIC program to POKE the program into memory.

ATUG (ASM/TED Users Group)
c/o Brent Anderson
200 S. Century
Rantoul, IL61866
217 893 4577

Editor Command List

(As of January 4, 1982)

This program adds commands to Basic–Aid that allow it to be a CBM Assembler Editor. Note that this editor should be KILLed before running the assembler because it uses some of the same memory locations in the second cassette buffer as the assembler does.

BASIC–AID Syntax: BASIC–AID

This command will disable these extra commands and initialize Basic–Aid only.

COLD Syntax: COLD

The COLD command will do a software rest of the PET. This reset is like a power–on restart.

FORMAT Syntax: FORMAT [line range]

The FORMAT command is used to print the text file in tabbed format like the assembler. For this function to work correctly you must type mnemonics in column two, or one space from labels. The line range is the same as for a LIST. The same keys as for Basic–Aid can be used for to hold, pause, and stop the listing.

GET Syntax: GET "filename" [,line #]

The GET command loads assembler text files from disk. This command is used to load source files into the editor and append to files already in memory. The file is input beginning at the line number given. GET starts numbering as 1000 by 10.

PUT Syntax: PUT "dr:filename" [,line range]
CPUT "dr:filename" [,line range]

The PUT command outputs source files to the disk for later assembly. PUT has the ability to output all or part of the memory resident file. The CPUT (for Crunch–PUT) command will remove extra spaces not in comments or ASCII text.

LOADER Syntax: LOADER "filename" [,hex offset]

The LOADER command does the same function as the separate loaders on the disk. The hex offset allows the object code to be stored at an address other than the assembled address. The same keys as in Basic–Aid can be used to hold, pause, and stop the listing.

Basic-Aid Assembler Editor Loader

```
100 t=0 : for i=1 to 5 : read ch(i) : next : read ch, ch
105 for i=1 to 5 : ch=0 : for j=0 to 255 : read x : ch=ch+x : next
110 print "checksum for block";i;"=";ch;
115 if ch=ch(i) then print "ok"
120 if ch<>ch(i) then print "error" : t=1
125 next i
130 if t<>0 then stop
135 restore : for i=1 to 5 : read ch(i) : next : print "loading. . ."
140 read lo, hi : for i=lo to hi : read x : poke i, x : next i
145 print "save $6b00 to $7000"
150 end
155 sys lo
160 rem checksums
165 data  31785, 31032, 32232, 33356, 21826
170 rem start and stop address
175 data  27392 , 28671
180 rem block 1
185 data  169,   0, 141, 136,   3, 173, 248, 111, 174, 249, 111, 224, 128, 176,   7, 133
190 data   52, 134,  53,  32, 233, 181, 162,  15, 189,  43, 112, 149, 112, 202,  16, 248
195 data  173, 250, 111, 174, 251, 111, 133, 122, 134, 123, 162,   0,  32,  53, 107,  32
200 data   35, 114,  76, 255, 179, 189,  73, 111, 240,   6,  32, 210, 255, 232, 208, 245
205 data   96,  76,  85, 113, 133, 128, 134, 129, 186, 189,   1,   1, 201,  15, 208, 241
210 data  189,   2,   1, 201, 180, 208, 234,  32, 123, 113, 144,  71, 165, 128,  16,   2
215 data  230, 119, 162,   0, 134, 124, 132, 130, 164, 119, 185,   0,   2,  56, 253, 184
220 data  111, 240,  22, 201, 128, 240,  22, 230, 124, 232, 189, 183, 111,  16, 250, 189
225 data  184, 111, 208, 228, 164, 130,  76,  85, 113, 232, 200, 208, 221, 132, 119, 104
230 data  104, 165, 124,  10, 170, 189, 231, 111,  72, 189, 230, 111,  72,  32, 121, 113
235 data   76, 112,   0, 104, 104, 165, 128,  32, 246, 184, 240,  47, 173, 136,   3, 240
240 data   42,  24, 165,  17, 109, 136,   3, 133,  96, 165,  18, 105,   0,  32,  63, 113
245 data  162,   0, 169,  32, 157, 111,   2, 232, 189,   0,   1, 240,   6, 157, 111,   2
250 data  232, 208, 245, 169,  32, 157, 111,   2, 232, 134, 158,  32, 132, 108,  76,  37
255 data  180,  32, 245, 108, 169,   1,  32,  14, 109,  32, 207, 255, 201,  13, 240,   9
260 data   32,  20, 109, 165, 150, 208,  32, 240, 240,  32,  45, 109,  32,  37, 109, 201
265 rem block 2
270 data    1, 208,   8,  32,  45, 109,  32,  42, 109, 208,   8,  32,  24, 109, 169,   0
275 data   32,  20, 109, 165, 150, 240, 205, 152,  32,  14, 109, 165,  92, 133,  42, 165
280 data   93, 133,  43,  32, 199, 109,  32,  54, 109, 169, 232, 133,  50, 169,   3, 133
285 data   51, 162,   0,  76, 159, 117, 162,   0,  44, 162, 255, 134, 131,  32, 191, 108
290 data   76,  75, 108, 169,  13,  32, 210, 255,  32,  24, 109,  32,  34, 109, 240,   7
295 data  166, 131,  32,  63, 109, 176,  10,  32, 199, 109,  76, 247, 114, 152,  32, 210
300 data  255,  32,  34, 109, 168, 240, 220, 201,  32, 208,   8, 138,  48, 239, 208, 241
305 data  232, 208, 234, 201,  59, 240,   4, 201,  39, 208,   2, 162, 255, 138,  48, 221
310 data  166, 131, 240, 217, 162, 255, 160,   0, 232, 189,   0,   2, 201,  32, 240, 248
315 data   36, 232, 189,   0,   2, 201,  48, 144,   4, 201,  58, 144, 244, 189,   0,   2
320 data  201,  32, 208,   1, 232, 189,   0,   2, 240,   9,  41, 127, 153,   0,   2, 232
325 data  200, 208, 242, 153,   0,   2, 200, 200, 153,   0,   2, 200, 200, 200,  96,  32
330 data   60, 245, 160,   0, 177, 218, 145,  42, 200, 196, 209, 208, 247, 162,   4, 189
335 data   68, 111, 145,  42, 200, 202, 208, 247, 132, 209, 165,  42, 166,  43, 133, 218
340 data  134, 219,  32, 118,   0, 240,   3,  32, 245, 190,  32,  52, 116,  32, 212, 109
345 data  162,   2,  76, 201, 255,  32,  60, 245,  32, 118,   0, 240,   3,  32, 245, 190
```

```
350 rem block 3
355 data  32,246,184, 32,163,181, 32,212,109,162,  2, 76,198,255, 32, 17
360 data 109, 32, 20,109,160,  0,145, 92,230, 92,208,  2,230, 93, 96, 32
365 data  34,109, 32, 24,109,160,  0,177, 92, 96, 32, 45,109,165, 92,208
370 data   2,198, 93,198, 92, 96,165, 40,133, 92,165, 41,133, 93, 96, 32
375 data  34,109,133, 64, 32, 34,109, 76,102,116, 32, 52,116,208,  3, 32
380 data  24,109, 32,223,186, 32,225,255, 32, 20,113, 32, 34,109,240,  5
385 data  32, 63,109,176,  3, 76,255,179,166, 64,165, 65, 32,131,207,169
390 data  32, 32,210,255,162,  0, 32, 34,109,240,212,201, 59,240, 56,201
395 data  32,240,  6, 32,210,255,232,208,237,224,  7,176,  6, 32,210,255
400 data 232,208,246, 32, 34,109,240,183,201, 32,240,247, 32,210,255,232
405 data  32, 34,109,240,170,201, 59,208,243, 72,169, 32,224, 23,176,  6
410 data  32,210,255,232,208,246,104, 32,210,255, 32, 34,109,240,144,208
415 data 246,108,252,255, 76,  0,112, 32,204,255,169,  1, 32,226,242,169
420 data   2, 76,226,242, 32,231,255,166,209,240, 99,134,  9,169,  1,133
425 data 210,169,  8,133,212,169, 15,133,211,169,  0,133,209, 32, 99,245
430 data  32,204,255,165,  9,133,209,169,  2,133,210,169,  8,133,212,169
435 rem block 4
440 data   2,133,211, 32, 99,245,162,  1, 32,198,255, 32,207,255,133,134
445 data  32,207,255,133,135,  5,134,201, 48,240, 25,166,134,165,135, 32
450 data  49,215, 32,207,255, 32,210,255,201, 13,208,246, 32,199,109,104
455 data 104, 76,255,179, 32,207,255,201, 13,208,249, 76,204,255,169,255
460 data 133, 55, 76,  0,191, 32, 60,245, 32,118,  0, 32,249,110,162,  0
465 data 134,132,134,133, 32,212,109,162,  2, 32,198,255, 32, 20,113,201
470 data 239,240,110, 32,207,255,166,150,208,109, 32,210,255,201, 59,208
475 data 235,160,  0,132, 31,132, 32, 32,226,110,240, 52,133,131, 32, 74
480 data 113, 32, 46,111, 32,239,110, 32, 39,111, 24,165,251,101, 96,133
485 data 251,165,252,101, 95,133,252, 32,226,110,145,251,209,251,208, 43
490 data 200, 32, 46,111,198,131,208,239, 32,239,110, 32, 56,111,240,172
495 data  32,239,110, 32, 39,111,165,251,197, 31,208, 24,165,252,197, 32
500 data 208, 18, 32,239,110, 32, 56,111, 76,215,110,162, 81, 44,162, 95
505 data  44,162, 60, 44,162, 42, 44,162, 68, 32, 53,107, 32,199,109, 76
510 data 255,179, 32, 99,215, 72,166,150,208,237, 32, 34,215,104, 96, 32
515 data  84,215,166,150,208,225, 76, 23,215,240, 33, 32,245,190,240, 28
520 rem block 5
525 data  32, 28,111, 32,141,215,162,  4,  6, 96, 38, 95,202,208,249,  5
530 data  96,133, 96, 32,112,  0,240, 14,198, 97,208,231,162,  0,134, 95
535 data 134, 96,162,  4,134, 97, 96,165,251, 32, 46,111,165,252, 24,101
540 data  31,133, 31,144,  2,230, 32, 96,165,251,197, 31,208,144,165,252
545 data 197, 32,208,138, 96, 87, 44, 83, 44,147, 42, 42, 42, 32, 67, 79
550 data  77, 77, 79, 68, 79, 82, 69, 32, 65, 83, 83, 69, 77, 66, 76, 69
555 data  82, 32, 69, 68, 73, 84, 79, 82, 32, 52, 46, 48, 32, 42, 42, 42
560 data  13, 13,  0, 13, 66, 65, 68, 32, 82, 69, 67, 79, 82, 68, 32, 67
565 data  79, 85, 78, 84,  0, 13, 66, 82, 69, 65, 75, 13,  0, 13, 69, 78
570 data  68, 32, 79, 70, 32, 76, 79, 65, 68,  0, 13, 78, 79, 78, 45, 82
575 data  65, 77, 32, 76, 79, 65, 68,  0, 13, 67, 72, 69, 67, 75, 83, 85
580 data  77, 32, 69, 82, 82, 79, 82,  0, 67, 79, 76,196, 67, 80, 85,212
585 data  70, 73, 78,196, 67, 72, 65, 78, 71,197, 70, 79, 82, 77, 65,212
590 data  71, 69,212, 80, 85,212, 66, 65, 83, 73, 67, 45, 65, 73,196, 76
595 data  79, 65, 68, 69,210,  0,192,109, 53,108, 19,115, 19,115, 73,109
600 data 224,107, 56,108,195,109, 68,110,  0,107, 68,107, 48, 49, 48, 52
```

The Commodore 64 Joystick Ports

Greg Beaumont
Ingersoll, Ont.

Anybody trying to use a joystick on the C64 may have found it a bit difficult. The major problem is the ports used for the joysticks are also used in keyboard decoding. So I decided to write a joystick routine that would suppress the keyboard scanning while reading the joystick values.

The joysticks ports are the parallel ports found in CIA 1 (6526) at $DC00 (56320 decimal). Each joystick requires 5 bits; four bits for the four major directions and one for the firebutton (see fig 1).

The sample program is self documenting. The possible values returned for the X direction are –1 (left), 0 (center), 1 (right), and for the Y are –1 (up), 0 (center), 1 (down). When a firebutton is pressed either a zero (not pressed) or non zero (pressed) value will be returned from the subroutine at 63000.

```
100 rem sample joystick program
110 sx=0 : sy=0 : print "■"
115 input "select joystick port (1 or 2)" ;p$
120 if p$<>"1" or p$<>"2" then 115
200 gosub 63000 : rem get joystick values
300 if p$="1" then sx=sx+ax : sy=sy+ay
305 if p$="2" then sx=sx+bx : sy=sy+by
310 if sx<0 or sx>40 then sx=0
320 if sy<0 or sy>40 then sy=0
400 poke 1024 + sx + 40*sy, 160
500 goto 200
```

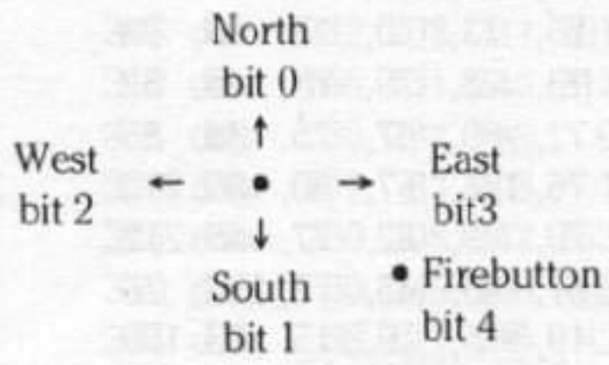
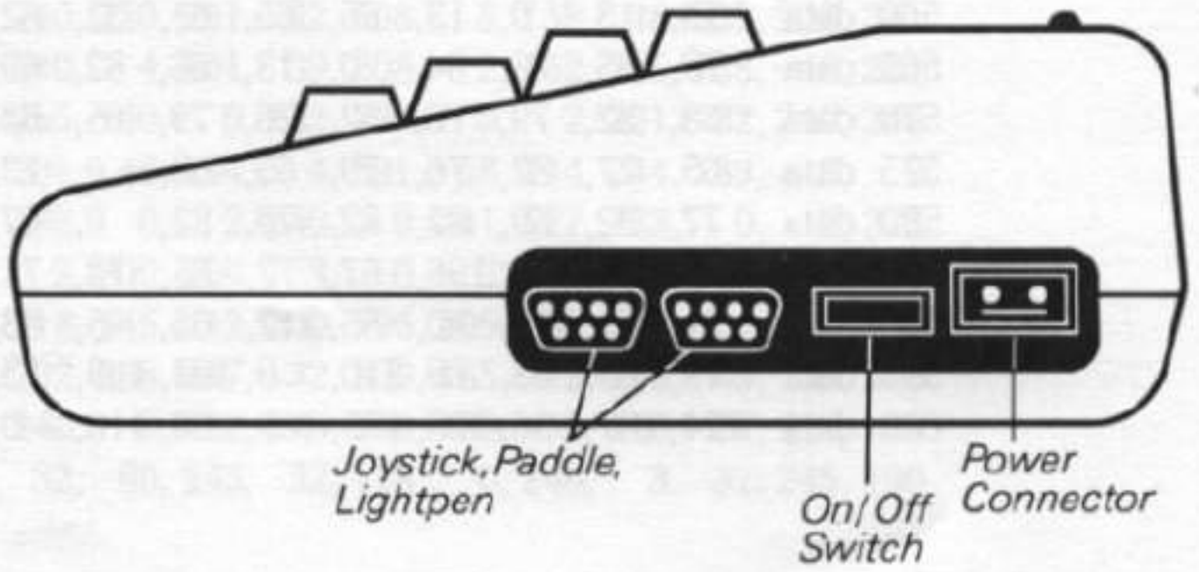


Fig. 1



```
63000 rem " Joystick routines for C64 by Greg Beaumont Ingersoll
63001 rem ax  : x value for joystick port 1
63002 rem ay  : y value for joystick port 1
63003 rem fa  : non zero if firebutton 1 is pressed
63004 rem bx  : x value for joystick port 2
63005 rem by  : y value for joystick port 2
63006 rem fb  : non zero if firebutton 2 is pressed
63010 if ft=0 then dim j(1,10) : for j=0 to 10 : read j(0,I), j(1,I) : next : ft=1
63015 cia = 13*4096 + 12*256                    : rem set pointer to cia 1 (joystick ports)
63020 poke cia+13, 127                          : rem disable irq interrupt
63030 poke cia+2, 0 : poke cia+3, 0             : rem set both joy ports to input mode
63040 a = 255-peek(cia) : b = peek(cia+1)       : rem get raw joystick values
63050 fa = a and 16 : fb = b and 16             : rem set fire button status
63060 ax = j(0, a and 15) : ay = j(1, a and 15) : rem set joystick port 1 values
63070 bx = j(0, b and 15) : by = j(1, b and 15) : rem set joystick port 2 values
63080 poke cia+2, 255                           : rem restore regular values
63090 poke cia+13, 255                          : rem restore irq interrupts
63100 return
63999 data 0, 0, 0, -1, 0, 1, 0, 0, -1, 0, -1, -1, -1, 1, 0, 0, 1, 0, 1, -1, 1, 1
```


COMMODORE USERS

Join the largest, active Commodore users group in North America and get—

— Access to club library of over 3000 free programs.

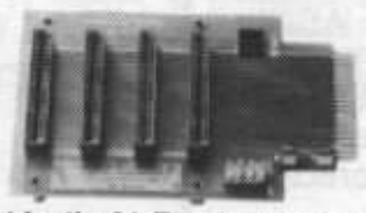
— Informative club newsletter.

— The latest information about the PET, CBM, VIC, Super-PET and Commodore-64.

Send $20.00 ($30.00 overseas) for Associate Membership to:

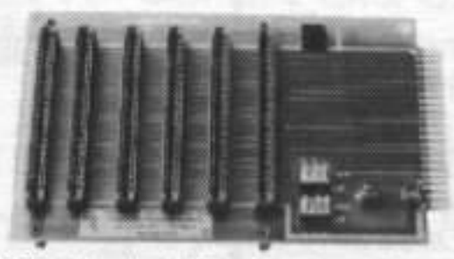
Toronto Pet Users Group
Department "M"
381 Lawrence Avenue West
Toronto, Ontario, Canada M5M 1B9

VIC-20 and CBM 64 EXPANDER BOARDS



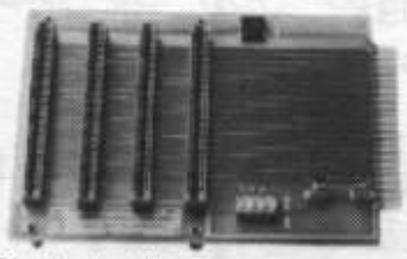
4 Slot for the 64. Toggle switches and reset switch.
P/N C64 $69.95

PTI offers the finest selection of expander boards available for the VIC-20 and CBM 64. The design features, quality construction, and competitive prices make any of them an exceptional value. New products are being added monthly, so write for complete catalog.

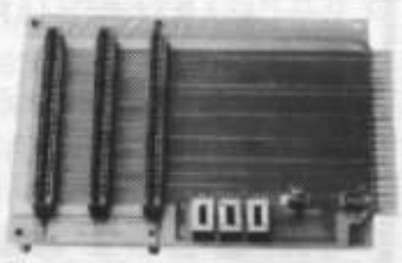


6 Slot for the VIC. Toggle switches and reset switch.
P/N V36 $79.95

Slot for the VIC. No switches, reset, or fuse.
P/N V13 $49.95



4 Slot for the VIC. Toggle switches and reset switch.
P/N V24 $69.95



3 Slot for the Vic. Slide switches, no reset switch.
P/N V23 $59.95

PRECISION TECHNOLOGY, INC.
COMPUTER PRODUCTS DIVISION
P.O. BOX 15454
SALT LAKE CITY, UTAH 84115
(801) 487-5266

See your dealer, or place your order direct

VISA-M/C-CHECK-COD

CBM/PET INTERFACES

The Connecting Links

CmC provides the link to increase your computer's functional ability. The following models come complete with case and cables and have a 30 day money back trial period.

PRINTER ADAPTERS
— addressable-switch selectable upper/lower, lower/upper case
— works with BASIC, WORDPRO, VISICALC and other software
— IEEE card edge connector for connecting disks and other peripherals to the PET
— power from printer unless otherwise noted

RS-232 SERIAL ADAPTER — baud rates to 9600 — power supply included
MODEL-ADA1450a $149.00

CENTRONICS/NEC PARALLEL ADAPTER — Centronics 36 pin ribbon connector
MODEL-ADA1600 . . . . . .$129.00

CENTRONICS 730/737/739 PARALLEL ADAPTER — 40 pin card edge connector
MODEL-ADA730. . . . . .$129.00

EPSON MX-70 PARALLEL ADAPTER — handles graphics — BASIC 4.0 required
MODEL-ADAX70. . . . . .$129.00

COMMUNICATIONS ADAPTER — serial & parallel ports — true ASCII conversion — baud rates to 9600 — half or full duplex — X-ON, X-OFF — selectable carriage return delay — 32 character buffer — centronics compatible — power supply included
MODEL-SADI $295.00

ANALOG TO DIGITAL CONVERTER — 16 channels — 0 to 5.12 volt input voltage range — resolution 20 millivolts per count — conversion time less than 100 microseconds per channel
MODEL-PETSET1 . . . . . .$295.00

US Dollars Quoted
$5.00 Shipping & Handling
MASTERCHARGE/VISA

All prices & specifications subject to change without notice

MENTION THIS MAGAZINE WITH ORDER AND DEDUCT 5% FROM TOTAL

IN THE USA order from:
Connecticut microComputer, Inc.
Instrument Division
36 Del Mar Drive
Brookfield, CT 06804
203-775-4595 TWX: 710 456-0052

IN CANADA order from:
Batteries Included, Ltd.
71 McCaul Street
F6 Toronto, Canada M5T2X1
(416) 596-1405

Dealer Inquiries Invited

64 ** ALL NEW!!! ** 64

SOFTWARE FOR COMMODORE 64

WORD-PAC $74.95
Print up to 99 pages of text.
Automatic tabbing/Centering/Underlining.
Copy Lines/Merge/Plus More!
Coded in Machine language.

CALC-PAC $74.95
Interface-Compatible with WORD-PAC & DATA-PAC.
Coded in our own Unique Spread-Sheet language.
User-Friendly Mathematical Applications.

DATA-PAC $39.95
Interface-Compatible with WORD-PAC & CALC-PAC.
User defined Formats/Search & Sorts.
Printer compatable.

EDITOR-PAC $69.95
Complete Programmer's Editor.
Auto-Number/Renumber including goto & gosub.
Program Merge/Global Search and Replace.
Plus Much More!

ASSEMBLER-PAC $59.95
Programmers take note!
Mnemonic format to Machine Language.
Link Modules/External references, More!

HOME-ACCOUNTANT $29.95
Checkbook with reconciliation routine.
Hard-Copy listing option.
Search and Review/Chart of Accounts.
Income and Expense.

ANNOUNCING . . .

The PCS/8064 Upgrade Module for the 64

On power-up the PCS/8064 provides:
80-column video output.
WORD-PAC word processing.
CALC-PAC spread sheet mathematics.
DATA-PAC data base system.
Exit to BASIC.
All Applications Interface-Compatible.

Plus Full line of Games/Home Software for 64

Dealer Inquiries Encouraged

Commodore 64 and 64 are trademarks of Commodore Business Machines

PACIFIC COAST SOFTWARE

3220 S. Brea Canyon Rd.
Diamond Bar, CA 91765
(714) 594-8210

218 S. Main/Box 147
LeSueur, MN 56058
(612) 665-6724

Mid-Eastern Distribution:
PERIPHERALS PLUS . . . . . (215) 687-8540
155 E. Lancaster Ave. - Wayne, Penn. 19087

New England Distribution:
OMICRON . . . . . (617) 769-6867
1416 Providence Highway - Norwood, Mass. 02062

Exclusive Canadian Distributor:
Computer Workshops Ltd. . . . . (416) 366-6192
465 King St. E. Unit 9 – Toronto, Ont. M5A 1L6

Z-RAM™ Opens Up The World Of CP/M™ To The Commodore Computer!

Z-RAM, a circuit board that fits inside the Commodore 4000 and 8000 computers, adds a Z-80A microprocessor and 64K of Random Access Memory (RAM), tripling the current maximum user memory!

Also provided with Z-RAM is the CP/M operating system, adding a new dimension to the Commodore computer by finally making the vast world of CP/M software available to its users.

With the addition of the Z-RAM board, Commodore computers will operate with 96K RAM and use both the 6502 and Z-80 processors. Several modes of operation are possible:

- With Z-RAM installed, all earlier versions of Commodore computers (regardless of memory capacity) can function as 32K machines. All current programs will run using 32K memory and the standard 6502 processor.
- Z-RAM permits the 6502 to use the full 96K of memory to operate the new expanded versions of Wordcraft Ultra, WordPro -Plus™ and other expanded Commodore programs.
- The Z-80A processor will operate in 64K memory and will run the industry standard CP/M operating system. WordStar™, Super-Calc™, and Accounting II Plus™ are only a sample of the fantastic CP/M programs now available to run on your Commodore computer!

Contact Your Nearest Commodore Dealer Today . . .
You'll Be So Glad You Did!

United States Distributor:
COMPUTER MARKETING SERVICES INC.
300 W. Marlton Pike, Suite 26
Cherry Hill, New Jersey 08002
(609) 795-9480

Canadian Distributor:
COMPUTER WORKSHOP
465 Kings Street East, Unit # 9
Toronto, Canada M5A 1L6
(416) 366-6192

Z-RAM is a trademark of Madison Computer
CP/M is a trademark of Digital Research

Advertising Index

Software

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Computer Marketing \Canadian Micro	Calc Result (spreadsheet prog.)			**IBC**				
Micro Applications	Master (programming aid)		IBC					
Pacific Coast Software	C64: Utility, Business, Games			**62**				
Precision Software	Superscript (wordprocessor)		61					
Wycor Business Systems	Provincial Payroll			**64**				

Hardware

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Computer Workshops \Computer Marketing	Z-RAM (CP/M board)	Madison		**63**				
Connecticut microComputer	PET/CBM Interface adapters			**62**				
Precision Technology	VIC20/C64 Expander Boards			**61**				
Richvale Telecommunications	C64 Link (IEEE adapter)			**IFC**				

Accessories

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Consultors Int'l	Stock Market With Your PC (book)		64					
"	A To Z Book Of Games		64					
"	Dynamics Of Money Mgmt (book)		64					
"	Invment. Analysis w/Your Micro (book)		64					
Leading Edge Inc.	Elephant Diskettes		BC					
Toronto PET Users Group	Membership info			**61**				

In The Next Issue Of

The Transactor

The Reference Issue
(The one you'll want to keep!)

- All CBM Memory Maps
- CBM ROM User Entry Points
- New Butterfield SuperChart
- 1982 Magazine Bibliography
- All CBM Port Pinouts
- Language Command Summary
- Plus loads more!

Keep On Transactin'...

PET/CBM

PROVINCIAL PAYROLL

Wycor Business Systems has developed a complete payroll system for Canada.

• Set up files for over 200 employees • Calculate and print payroll journal • Print cheques • Print monthly submission for Revenue Canada • Accumulate and print T-4s • Complete employee lists.

This system comes with full user documentation and tutorial disk.

Complete System $850.00
Manual only 25.00

Call collect (416) 444-3492 for information or contact your dealer.

WYCOR BUSINESS SYSTEMS LIMITED
170 THE DONWAY WEST, STE. 401,
DON MILLS, ONT. M3C 2G3

A More Powerful Planning
And Forecasting Tool Than Any Other On The Market . . .

Calc Result

A three-dimensional spread sheet
with multiple pages of 63 x 254 cells
which utilizes only the memory
in cells that are active

Produces Graphics (Histograms) on screen
and printer

Gives unlimited possibilities in each cell with
IF-THEN-ELSE with AND, OR and NOT-ELSE

Available now for 8032 and 8096 – *coming soon* for Commodore 64.

FOR FURTHER INFORMATION CONTACT YOUR NEAREST COMMODORE DEALER

OR

IN CANADA

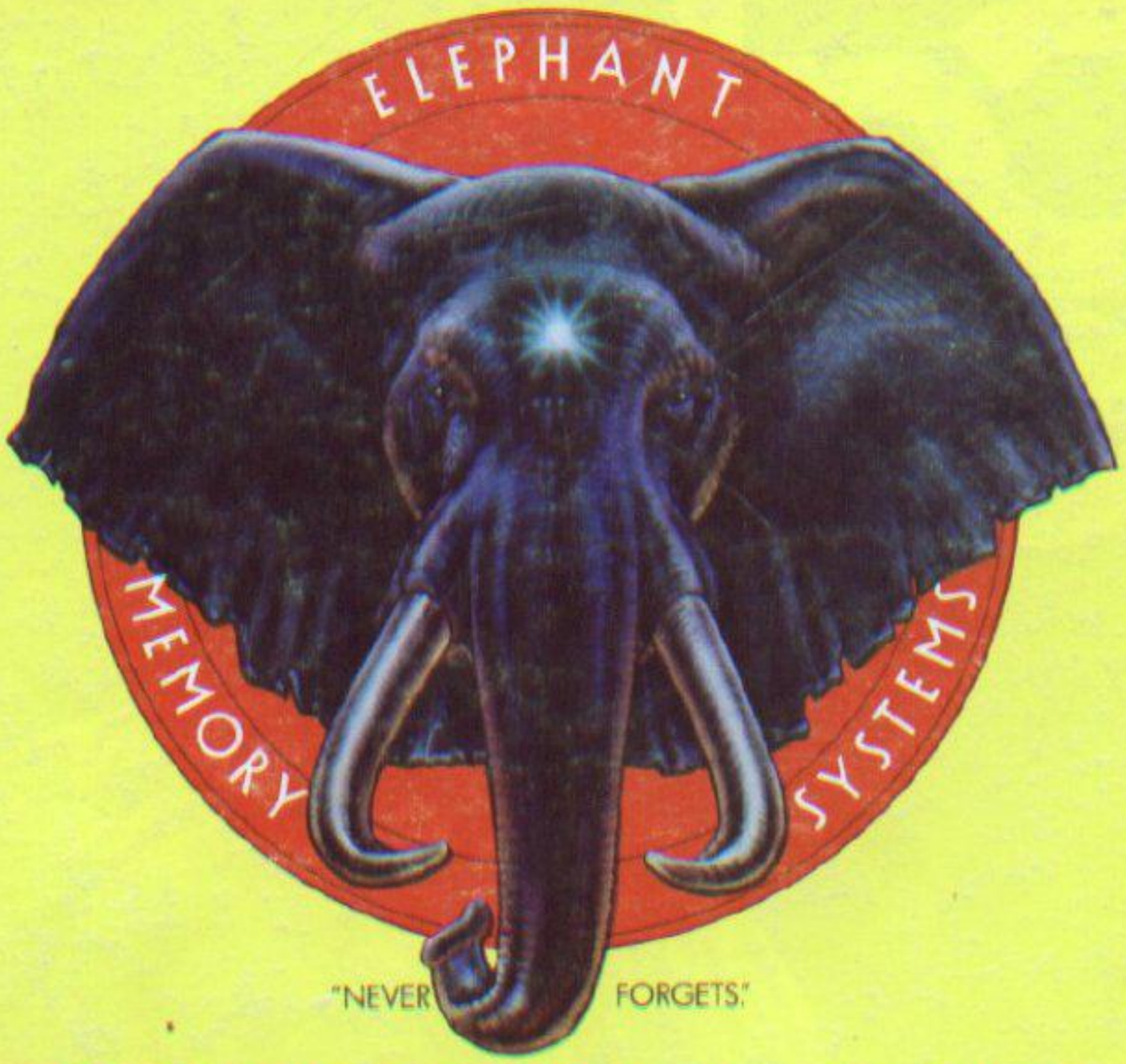
CANADIAN MICRO DISTRIBUTORS
500 Steeles Avenue
Milton, Ontario, Canada L9T 3P7
(416) 878-7277

IN THE U.S.A.

COMPUTER MARKETING SERVICES INC.
300 West Marlton Pike, Suite 26
Cherry Hill, New Jersey, U.S.A. 08002
(609) 795-9480

CALC RESULT *is a trademark of Handic Software ab*

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021

Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.